Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects

MAIRIELI WESSEL, University of São Paulo, Brazil

IGOR S. WIESE, Universidade Tecnológica Federal do Paraná, Brazil

IGOR STEINMACHER, Northern Arizona University, USA and Universidade Tecnológica Federal do Paraná, Brazil

MARCO A. GEROSA, Northern Arizona University, USA

Software bots are used to streamline tasks in Open Source Software (OSS) projects' pull requests, saving development cost, time, and effort. However, their presence can be disruptive to the community. We identified several challenges caused by bots in pull request interactions by interviewing 21 practitioners, including project maintainers, contributors, and bot developers. In particular, our findings indicate noise as a recurrent and central problem. Noise affects both human communication and development workflow by overwhelming and distracting developers. Our main contribution is a theory of how human developers perceive annoying bot behaviors as noise on social coding platforms. This contribution may help practitioners understand the effects of adopting a bot, and researchers and tool designers may leverage our results to better support human-bot interaction on social coding platforms.

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

Additional Key Words and Phrases: Software Bots, GitHub Bots, Human-bot Interaction, Open Source Software, Collaborative Development, Software Engineering

1 INTRODUCTION

Open Source Software (OSS) development is inherently collaborative, frequently involving geographically dispersed contributors. OSS projects often are hosted in social coding platforms, such as GitHub and GitLab, which provide features that aid collaboration and sharing, such as pull requests [56]. Pull requests facilitate interaction among developers to review and integrate code contributions. To alleviate their workload [17], project maintainers often rely on software bots to check whether the code builds, the tests pass, and the contribution conforms to a defined style guide [18, 24, 58]. More complex tasks include repairing bugs [35, 57], refactoring source code [66], recommending tools [3], updating dependencies [34], and fixing static analysis violations [4].

Authors' addresses: Mairieli Wessel, University of São Paulo, Brazil, mairieli@ime.usp.br; Igor S. Wiese, Universidade Tecnológica Federal do Paraná, Brazil, igor@utfpr.edu.br; Igor Steinmacher, Northern Arizona University, USA, Universidade Tecnológica Federal do Paraná, Brazil, igor.steinmacher@nau.edu; Marco A. Gerosa, Northern Arizona University, USA, marco.gerosa@nau.edu.

The introduction of bots aims to save cost, effort, and time [53], allowing maintainers to focus on development and review tasks. However, new technology often brings consequences that counter designers' and adopters' expectations [20]. Developers who *a priori* expect technological developments to lead to performance improvements can be caught off-guard by *a posteriori* unanticipated operational complexities and collateral effects [65]. For example, the literature has shown that although the number of human comments decreases after the introduction of bots [63], many developers do not perceive this decrease [62]. These collateral effects and the misalignment between the preferences and needs of project maintainers and bot developers can cause expectation breakdowns, as illustrated by a developer: "*Whoever wrote <bot-name> fundamentally does not understand software development.*"[1] Moreover, as bots have become new voices in developers' conversation [35], they may overburden developers who already suffer from information overload when communicating online [40]. On an abandoned pull request, a maintainer complained about the frequency of action of a bot: "*@<bot-name> seems pretty active here [...].*"[2] Changes the technology provokes in human behavior may cause additional complexities [36]. Therefore, it is important to assess and discuss the effects of a new technology on group dynamics; yet, this is often neglected when it comes to software bots [41, 53].

Considering developers' perspectives on the overall effects of introducing bots, designers can revisit their bots to better support the interactions in the development workflow and account for collateral effects. So far, the literature presents scarce evidence, and only as secondary results, of the challenges incurred when adopting bots. Investigating the usage of the *Greenkeeper* bot, Mirhosseini and Parnin [34], for example, report that maintainers are overwhelmed by bot pull request notifications interrupting their workflow. According to Brown and Parnin [3], the human-bot interaction on pull requests can be inconvenient, leading developers to abandon their contributions. This problem may be especially relevant for newcomers, who require special support during the onboarding process due to the barriers they face [49, 50]. Newcomers can perceive bots' complex answers as discouraging, since bots often provide a long list of critical contribution feedback (e.g., style guidelines, failed tests), rather than supportive assistance.

We extend previous work by delving into the challenges incurred by bots on social coding platforms. To do so, we investigate the challenges bots bring to the pull request workflow from the perspective of practitioners. Specifically, our work investigates the following research question:

RQ. *What interaction challenges do bots introduce when supporting pull requests?*

To answer our research question, we qualitatively analyzed data collected from semi-structured interviews with 21 practitioners, including OSS project maintainers, contributors, and bot developers who have experience interacting with bots on pull requests. After analyzing the interviews, we validated our findings through member-checking.

While participants commend bots for streamlining the pull request process, they complain about several challenges they introduce. Some of the challenges include annoying bot behaviors such as verbosity, too many actions, and unrequested or undesirable tasks on pull requests, which are often perceived as noise. Since noise emerged as a central theme in our analysis, we further theorize about it, grounded in the data we collected.

Our work contributes to the state-of-the-art by (i) identifying a set of challenges incurred by the use of software bots on the pull requests' workflow and (ii) proposing a theory about how noise introduced by bots disrupts developers' communication and workflow. We present a set of 17 general challenges that have not been reported in the literature. By gathering a comprehensive set of challenges incurred by bots, our findings complement the previous literature, which presents

[1]https://twitter.com/mojavelinux/status/1125077242822836228
[2]https://github.com/facebook/react/pull/12457#issuecomment-413429168

scarce and diffuse challenges, reported as secondary results. Our contributions support practitioners in understanding, or even anticipating, the impacts that adopting a bot may have on their projects. Researchers and tool designers may also leverage our results to enhance bots' communication design, thereby better supporting human-bot interaction on social coding platforms.

2 BACKGROUND

According to Storey and Zagalsky [53], a software bot is *"a conduit or an interface between users and services, typically through a conversational user interface"*. In the following, we provide more details about the existing literature related to the challenges of using software bots, especially on social coding platforms.

2.1 Challenges of bots in online communities

Software bots have been extensively studied in the literature of different domains, including social media [1, 11, 45, 67], online learning [13, 26, 39], and Wikipedia [6, 12]. Despite the widespread adoption of bots in different domains, the interaction between computers and humans still presents challenges [7, 59, 69]. For example, Zheng et al. [68] describe how although editors appreciate Wikipedia bots for streamlining knowledge production, they complain that the bots create additional challenges. To circumvent some of these challenges, Wikipedia established rigid governance roles [37]. Bots need to contain the string "bot" in their username, have a discussion page that clearly describes what they do, and can be turned off by any member of the community at any time.

In recent years, software bots have also been proposed to support collaborative software engineering, encompassing both technical and social aspects of software development activities [28]. According to Lebeuf et al. [27], bots provide an interface with additional value on top of the software service's basic capabilities. Interviewing industry practitioners, Erlenhov et al. [10] found that bots cause interruption and noise, trust, and usability issues.

2.2 Challenges of bots on social coding platforms

In the scope of social coding platforms, Wessel et al. [61] conducted a study to characterize the bots that support pull requests on GitHub. Their results indicate that bot adoption is widespread in OSS projects and are used to perform a variety of tasks on pull requests. The authors also report some challenges of using bots on pull requests. Several contributors complained about the way the bots interact, saying that the bots provide non-comprehensive or poor feedback. In contrast, others mentioned that bots introduce communication noise and that there is a lack of information on how to interact with the bot.

Certain bots have been studied in detail, revealing challenges and limitations of their interventions in pull requests. For example, while analyzing the *tool-recommender-bot*, Brown and Parnin [3] report that bots still need to overcome problems such as notification workload. Mirhosseini and Parnin [34] 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 [43] conducted a case study on how developers perceive and work with *mention bot*. The results show that this bot has saved developers' efforts; however, it may not meet the diverse needs of all users. For example, while project owners require simplicity and stability, contributors require transparency, and reviewers require selectivity. Despite its potential benefits, results also show that developers can be bothered by frequent review notifications when dealing with a heavy workload.

Although several bots have been proposed, relatively little has been done to evaluate the state of practice. Furthermore, although some studies focus on designing and evaluating bot interactions, they do not draw attention to potential problems introduced by these bots at large. According to Brown and Parnin [3], bots still need to enhance their interaction with humans. Responding to

this gap, we complement the findings from previous works by delving deeper into the challenges that bots bring to interactions on social coding platforms. This study takes a closer look at how practitioners interact with bots and what challenges they face. Also complementing the previous literature, we discuss how noise is characterized in terms of its impacts and how developers have attempted to handle it.

3   RESEARCH DESIGN

The main goal of this study is to identify challenges caused by bots on pull request interactions. To achieve this goal, we conducted a qualitative study of responses collected from semi-structured interviews. Figure 1 shows an overview of the research design employed in this study.
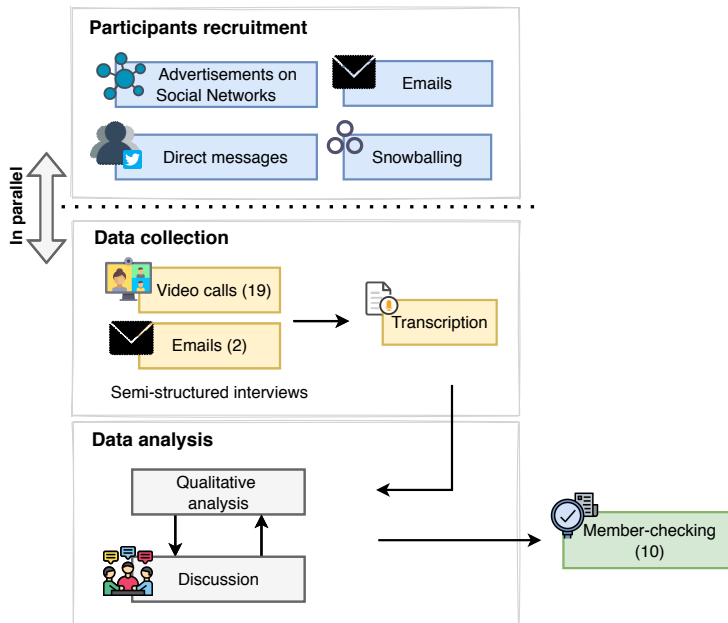

Fig. 1. Research Design Overview

3.1   Participants recruitment

We recruited participants from three different groups: (i) project maintainers, (ii) project contributors, and (iii) bot developers. Participants were expected to have experience contributing to or maintaining projects that use bots to support pull request activities. We adopted four main strategies to invite participants: (i) advertising on Twitter, (ii) direct messages, (iii) emails, and (iv) snowballing. Besides the broad advertisement posted on Twitter, we also manually searched for users that had posted about GitHub bots or commented on posts related to GitHub bots. During this process, we sent direct messages to 51 developers. In addition, we used the GitHub API to collect a set of OSS projects that use more than one bot. After collecting a set of 225 GitHub repositories using three or more bots, we sent 150 emails to maintainers and contributors whose contact information was publicly available. In addition, we asked participants to refer us to other qualified participants. We continued inviting participants as the data unveiled new relevant information. The participants received a 25-dollar gift card as a token of appreciation for their time.

Table 1. Demographics of interviewees

Participant ID	SD Experience (years)	Experienced with bots as			Location
		Maintainer	Contributor	Bot developer	
P1	9	✓	✓		Europe
P2	2	✓			South America
P3	20	✓	✓	✓	North America
P4	10	✓		✓	North America
P5	12	✓	✓	✓	North America
P6	4	✓	✓		North America
P7	10	✓			North America
P8	10	✓*			North America
P9	14	✓		✓	Europe
P10	12	✓	✓		South America
P11	5		✓		Europe
P12	20	✓		✓	North America
P13	25	✓			North America
P14	25	✓			Europe
P15	13		✓		North America
P16	20	✓	✓		Europe
P17	8	✓		✓	North America
P18	5		✓		Europe
P19	5	✓	✓		North America
P20	4	✓			Europe
P21	11	✓			Europe

* Also described himself as a *bot evangelist*

As a result of our recruitment, we interviewed 21 participants—identified here as P1 – P21. Table 1 presents the demographics of our interviewees. Their experience with software development ranges from 3 to 25 years (≃ 12 years on average). Participants are geographically distributed across North America (≈53%), Europe (≈38%), and South America (≈10%). Three interviewees are project contributors who have interacted with bots when submitting pull requests to open-source projects. All the other interviewees (18) maintain projects that use bots to support pull request activities. Besides their experience as project maintainers, seven of them also have experience in contributing to other projects that use bots. Six maintainers have experience building bots. One of the maintainers also described himself as a "bot evangelist."

Additionally, participants have experience with diverse types of bots, including project-specific bots, dependency management bots (e.g., Dependabot, Greenkeeper), code review bots (e.g., Codecov, Coveralls, DeepCode), triage bots (e.g., Stale bot), and welcoming bots (e.g., First Timers bot). Their experience ranges from interacting with 1 to 6 bots (≃ 2 bots on average), encompassing a total of 24 different bots. Further, bot developers develop between 1 and 3 bots (≃ 1 on average). For confidentiality reasons, we do not report either the bots used by each participant or their projects.

3.2 Semi-structured interviews

To identify the challenges, we conducted semi-structured interviews, which comprised open- and closed-ended questions that enabled interviewers to explore interesting topics that emerged during the interview [21]. By participants' requests, 2 interviews (P1 and P20) were conducted via email. The other 19 interviews were conducted via video calls. We started the interviews with a short explanation of the research objectives and guidelines, followed by demographic questions. The rest of the interview script focused on three main topics: (i) experience with GitHub bots, (ii) main challenges introduced by the bots, and (iii) the envisioned solutions to those challenges. The detailed interview script is publicly available[3]. Each interview was conducted remotely by the first author of this paper and lasted, on average, 46 minutes.

3.3 Data analysis

We qualitatively analyzed the interview transcripts, performing open and axial coding procedures [51, 55] throughout multiple rounds of analysis. We started by applying open coding, whereby we identified challenges brought by the interaction, adoption, and development of bots. To do so, the first author of this paper conducted a preliminary analysis, identifying the main codes. Then, we discussed the coding in weekly hands-on meetings, aiming to increase the reliability of the results and mitigate bias [42, 55]. In these meetings, all the researchers revisited codes, definitions, and their relationships until reaching an agreement. Afterwards, the first author further analyzed and revised the interviews to identify relationships between concepts that emerged from the open coding analysis (axial coding). Then, the entire group of researchers discussed the concepts and their relationships during the next weekly meeting. During this process, we employed a constant comparison method [14], wherein we continuously compared the results from one interview with those obtained from the previous ones. The entire analysis lasted eight weeks and each weekly meeting lasted from 1 to 2 hours.

For confidentiality reasons, we do not share the interview transcripts. However, we made our complete code book publicly available[3]. The code book includes the code names, descriptions, and examples of quotes for all categories.

3.4 Member-checking

As a measure of trustworthiness, we member-check our final interpretation of the theory about noise introduced by bots with the participants. The process of member-checking is an opportunity for participants check particular aspects of the data they provided [32]. According to Charmaz [5], member-checking entails "*taking ideas back to research participants for their confirmation.*" Such checks might occur through returning emerging research findings or a research report to individual participants for verification of their accuracy.

We contacted our 21 participants via email. In the email, we included the theory, followed by a short description of the concepts and their relationships. Participants could provide feedback by email or through an online meeting. Ten participants provided their feedback: P2, P3, P4, P7, P13, P16, P18, and P20 provided a detailed feedback by email, whereas P10 and P12 scheduled an online meeting, each lasting about 20 minutes.

The participants who gave feedback agreed with the accuracy of the theory about noise introduced by bots. P4, an experienced bot developer, described our research in a positive light, saying it "*captures the problem of writing an effective bot.*" The participants suggested a few adjustments. For instance, P12 recommended including another countermeasure to avoid noise ("*re-designing the bot*"). We addressed the feedback by including this countermeasure to our theory. Additional comments from member-checking can be found in our code book, tagged as "*from member-checking*".

4 FINDINGS

In this section, we present the challenges reported by the participants, as well as a theory focused on explaining the reasons and effects of the noise caused by bots on pull requests.

4.1 Challenges incurred by bots

The interviewees reported social and technical challenges related to the development, adoption, and interaction of bots on pull requests. Figure 2 shows a hierarchical categorization that summarizes these challenges. We added a graphical mark (⌨) in the hierarchical categorization to identify challenges that have been also identified by previous work, described in Section 2. In summary, we

---

[3]https://zenodo.org/record/4088774

found 25 challenges, organized into three categories (development challenges, adoption challenges, and interaction challenges) and several sub-categories. In the following, we present these three main categories of challenges, focusing on the 12 challenges related to the human-bot interaction on pull requests, since they strongly align with the challenges posed by the theory about noise introduced by bots. We describe the categories in **bold**, and provide the number of participants we assigned to each category (in parentheses).
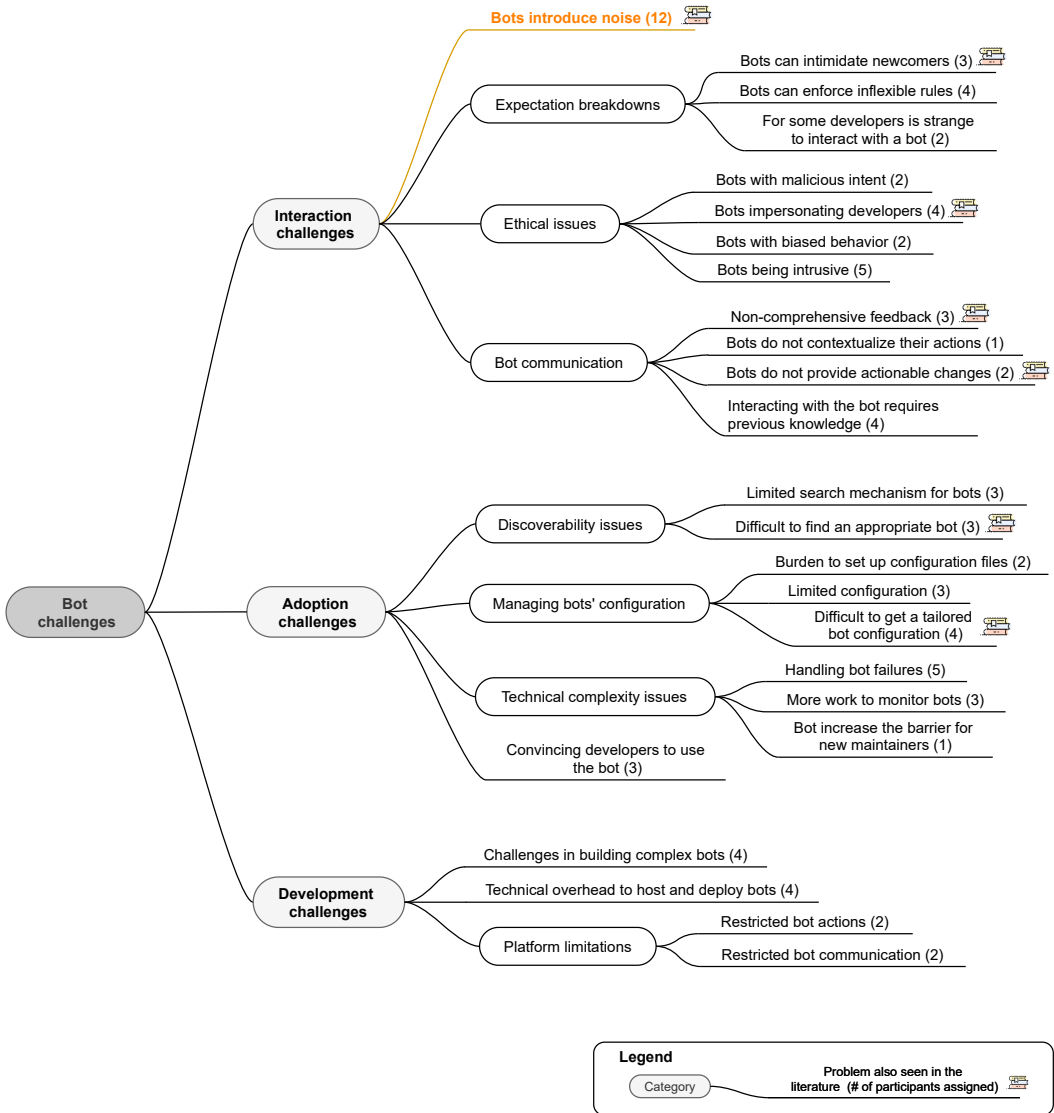

Fig. 2. Hierarchical Categorization of Bot Challenges

4.1.1 Bot interaction challenges. Concerning the human-bot interaction on pull requests, the most recurrent and central challenge according to our analysis is that **bots introduce noise** (12) to the

human communication and development workflow. We discuss the results that are specific to noise in Section 4.2, where we describe the proposed theory about noise caused by bots.

With regards to bot communication, we unveiled four challenges. We noticed that **interacting with the bot requires other technical knowledge** (4) not related to the bot. As a consequence, for example, developers might trigger a bot by accident or even misuse the bot capabilities. P5 explained that some developers are not aware of how auto-merging pull requests works on GitHub, which leads contributors to misuse the bot that they developed to support this functionality. This happens due to the way bots are designed to interact. As described by P4, bots perform tasks and need to communicate with humans; however, they do not understand the context of what they are doing. Therefore, we observed that **bots do not contextualize their actions** (1) and sometimes provide **non-comprehensive feedback** (3). In these cases, when a bot message is not clear enough, developers "*[...] need to go and ask a human for clarity.*" [P17], which may generate more work for both contributors and maintainers. In addition, bots **do not provide actionable changes** (2) for developers, meaning that some bots' messages and outcomes are so strict that do not guide developers on what they should do next to accomplish their tasks. According to P8 "*it is great to see 'yes' or 'no', but if it is not actionable, then it is not useful [...]*".

Since OSS developers come from diverse cultures and backgrounds, their cultural differences and previous experiences influence how they interact with and react to a bot's action. We observed three main challenges related to developers' *expectation breakdowns* when interacting with bots on pull requests. First, bots can **enforce inflexible rules** (4). These rules are commonly imposed by a specific community and evidenced by bot actions. For example, P7 mentioned: "*so, the biggest complaints we have gotten are that our lint rules and tests are too strict. And of course, the bot enforces that.*" In addition, we found that the way these inflexible rules are interpreted can vary based on developers' expectations. P7 suggested that the bots' "*social issues largely come down to a bot being inflexible and not meeting somebody's expectations*". Another complaint refers to **bots intimidating newcomers** (3). For new contributors to an OSS project, interacting with a bot that they have never seen or heard of before might be confusing, and the newcomers might feel intimidated, as stated by P12: *"If you're new to a project, then you might not be expecting bots, right? So, if you don't expect it, then that could be confusing".* Furthermore, some developers might find it **strange to interact with a bot** (2), as mentioned by P12: "*'Hey, I'm here to help you' [...] for some people, it is still quite strange, and they are quite surprised by it.*" Further, P5 also mentioned that receiving "thanks" from a non-human feels less sincere.

From an ethical perspective, we identified four challenges. Five participants reported **bots as intrusive** (5). For them, an intrusive bot is a bot that modifies commits and pull requests: "*let's say you have a very large line of code and the bot goes there and breaks that line for you. It is intrusive because it is changing what the developers did.*" [P21]. Another example of intrusive bots are those created for spamming repositories. P4 mentioned the case of *Orthographic Pedant* bot.[4] This bot searches for repositories in which there is a typo, then creates a pull request to correct the typo. The biggest complaint about this bot is that the developers did not allow the bot to interact on their projects, as P4 explained: "*people want to have agency, they want to have a choice. [...] They want to know that they are being corrected because they asked to be corrected.*" In addition, **bots impersonating developers** (4) were also mentioned as a challenge by our interviewees. Two other ethical challenges reported during our interviews were **malicious intent** (2) and **biased behavior** (2). Bots with a malicious intent could "*manipulate developers actions*" [P9], for example, by including a security venerability into the source code by merging a pull request. Further, according to P9,

---

[4]https://github.com/thoppe/orthographic-pedant

as there is no criteria to verify the use of bots, they can have a *biased behavior* and represent the opinion of a particular entity (e.g., the enterprise who created the bot).

4.1.2   Bot adoption challenges. Participants also mentioned challenges related to the adoption of bots into their GitHub repositories. According to P4, the challenges of bot adoption begin with finding the right bot. Developers complain that it is **difficult to find an appropriate bot** (3) to solve their problems. As P4 explained, there is a **limited search mechanism for bots** (3). P6 added: "*In the [GitHub] marketplace, [...] I don't even know if there is a category for bots.*" If maintainers find an appropriate bot, they then have to deal with configuration challenges. First, it is **difficult to tailor configuration** (4) to fit the needs of a project. Even after maintainers spend the time needed to configure the bot, there is no way to predict what the bot will do once installed. In P10's experience, it is "*easy to install the bot with the basic configuration. However, it is not easy to adjust the configuration to your needs*". A related challenge is the **limited configuration** (3) settings provided by the bots. There are limited resources, for example, to integrate the bot into several projects at once. Some participants also mentioned the **burden to set up configuration files** (2): "*It is like a whole configuration file you have to write. That is a lot of work, right?*" [P4]. Maintainers also need to deal with technical complexity issues caused by bot adoption, such as **handling bots failures** (5). Due to bot instability, our interviewees also mentioned that there is **more work to monitor bots** (3) to guarantee that everything is working well. Another technical issue is that adopting a **bot increases the barrier for new maintainers** (1), who need to be aware of how each bot works on the project.

4.1.3   Bot development challenges. We also identified challenges related to bot development. Firstly, bot developers often face *platform limitations*, commonly due to **restricted bot actions** (2). As mentioned by P5: "*There are still a few things that just cannot be done with the [GitHub] API. So that's a problem that I face.*" The platform restrictions might limit both the extent of the bots' actions and the way bots communicate. Regarding the **restricted bot communication** (2), P4 stated that the platform ideally would provide additional mechanisms to improve it, since the only way bots communicate is through comments. Participants also reported **technical overhead to host and deploy a bot** (4). P13 identified the "*main trouble with bots right now is you have to host them.*" Therefore, when a developer has to maintain the bot itself (e.g., project-specific bots), it becomes an overhead cost, since "*the bot saves you time but it also costs time to maintain*" (P19). In addition, we found **challenges in building complex bots** (4). For example, P12, an experienced bot developer, reports that bots found in other projects "*just automate a single thing. We just have one bot that does everything. I think it is hard to build a bot that has a lot of capabilities.*"

> **Summary about challenges caused by bots.** We provided a hierarchical categorization of bot challenges and focused on the human-bot interaction challenges. We found 12 challenges regarding bot communication, expectations, and ethical issues. Among these challenges, we found noise as a recurrent and central challenge.

4.2   Theory about noise introduced by bots

As aforementioned, the most recurrent and central problem reported by our interviewees was the introduction of noise into the developers' communication channel. This problem was a crosscutting concern related to bots' development, adoption, and interaction in OSS projects. Figure 3 shows the high-level concepts and relationships that resulted from our qualitative analysis. Some interviewees complained about **annoying bot behaviors** such as verbosity, high frequency and timing of actions, and unsolicited actions. Interviewees also mentioned a **set of factors** that might *cause*

annoying behaviors. These behaviors are often *perceived as* **noise**. The noise introduction leads to information overload (i.e. notification overload, extra information for maintainers), which *disrupts* both **human communication** and **development workflow**. To handle the challenges provoked by noise, developers rely on **countermeasures**, such as re-configuring or re-designing the bot.
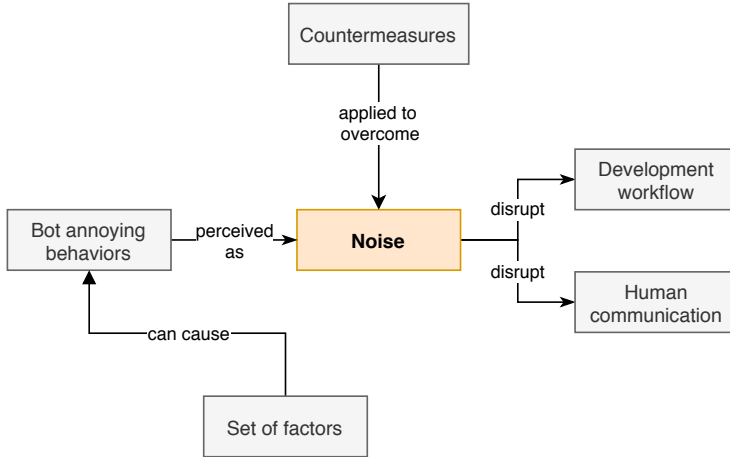

Fig. 3. High-level concepts and relationships of Bots' Noise Theory

In the following, we present in detail the theory about noise introduced by bots described in Figure 4. As previously, we include the concepts in **bold** face and the (sub-)categories in *italic*. We also provide the number of participants for each category (in parentheses).
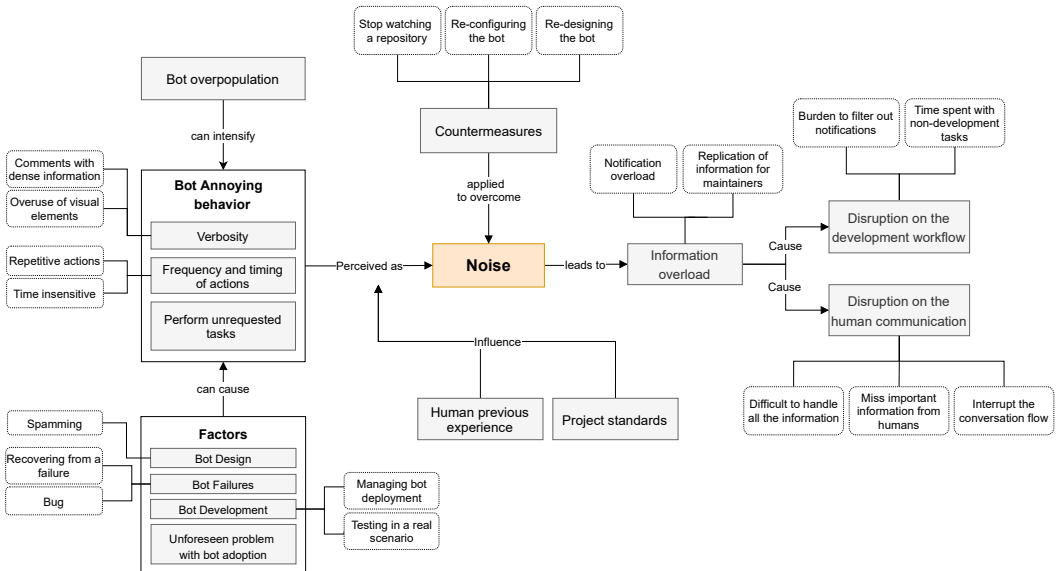

Fig. 4. Theory about noise introduced by bots

4.2.1 Bot annoying behaviors. Interviewees reported several annoying behaviors when bots interact on pull requests. The most recurrent one was the high **frequency and timing** of bots' actions (8). This includes the case in which the interviewees say that bots perform *repetitive actions*, such as creating numerous pull requests and leaving dozen of comments in a row. P6 explained: "*[...] is an automation that runs too frequently and then it keeps opening up all the pull requests that I do not need or want to.*" In addition, P9 mentioned complaints about the frequency of bot comments: "*sometimes we get comments like 'hey, bot comments too much to my taste'.*" Besides that, bot actions are usually *time insensitive*. Bots are designed to "*work all day long*" [P10] which might interrupt the developer at the wrong time. P3 offered an exemplary case of how a welcoming bot might be *time insensitive*: "*as long as, for example, the comment is immediately after I did a change [...] if it is in a second or two and I'm seeing the page I do not get a new notification. But if it happens three minutes later, and I left the page and suddenly I get the new notification and I think 'ah, this person has another question or something,' so I need to check it out and find out that this is a bot.*"

Another annoying behavior regards the bots' **verbosity** (5). Participants complained about bots providing comments with *dense information* "*in the middle of the pull request*" [P13], oftentimes *overusing visual elements* such as "*big graphics*"[P13]. In P19's experience, developers frequently do not like when "*[...] bots put a bunch of the information that they try to convey in comments instead of [providing] status hooks or a link somewhere.*" P17 reinforced this issue regarding: "*[...] a GitHub integration [bot] that posts these rules. Really dense and information rich elements to your pull requests. And I've seen it be a lot more distracting than it is helpful.*"

Another common annoying behavior regards the **execution of unrequested or undesirable tasks** (4) on pull requests. Participants mentioned that, due to external factors, or even due to the way bots have been designed to interact on pull requests, bots often perform tasks that were neither required nor desired by human developers. P6 described an issue caused by an external failure that impacted the bot interaction: "*Something went wrong with the release process. So, [the bot] opened up a bunch of different pull requests. And like some of them were a mistake. The other engineer that had to comment and be like, 'Hey, sorry, these were a mistake'.*"

To illustrate the described behaviors, we highlighted some examples cited by our participants and described in the state-of-the-practice. Figure 5a shows the case of a verbose comment, which included a lot of information and many graphical elements, inserted by a bot in the middle of a human conversation. In Figure 5b, we show a bot overloading a single repository with many pull requests, even if there were opened pull requests by the same bot. Finally, Figure 5c depicts a bot spamming a repository with an unsolicited pull request.
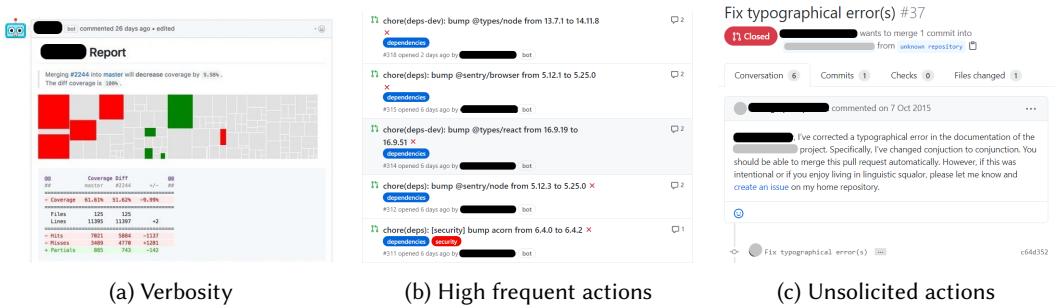

(a) Verbosity   (b) High frequent actions   (c) Unsolicited actions

Fig. 5. Examples of annoying behaviors from the state-of-the-practice

4.2.2   What might cause an annoying behavior? Several **factors** provoke annoying behaviors, sometimes by **bots' design** (4). Some bots are intentionally designed to *spam* repositories, as reported in the interaction challenges in Section 4.1.1. Other bots might demonstrate a certain behavior by default, as said by P19 when talking about a bot that reports code coverage: "*but by default, it also leaves a comment.*"

We also found unintended factors that might trigger annoying behaviors. For example, **bot failures** (4) might be responsible for triggering unsolicited tasks, or even increasing the frequency of bot actions. As shown in Section 4.1.3, handling bot failures is one of the challenges faced by project maintainers. According to P3, when the stale bot, which triages issues and pull requests, recovers from a failure, it posts all missed comments and closes all pull requests that need to be closed. As a consequence, it suddenly overloads both maintainers and contributors. As P7 describes: "*the only times I perceived our bots as noisy is when there is an obvious bug.*" Further, an **unforeseen problem with bot adoption** (3) also may result in unexpected actions or overloading maintainers with new information. Once the stale bot is installed, for example, it comments on every pull request that is open and no longer active. As P3 comments: "*this is what you want, but it is also a lot of noise for everyone who is watching the repository.*"

In addition, interviewees also mentioned issues during **bot development** (3) that might trigger an annoying behavior. As reported in the bot development challenges (Section 4.1.3), bot developers, for example, often face technical overhead costs to host and deploy bots. P7 reported that once they tried to upgrade the bot, it led to an "*edit war,*" resulting in the bot performing unsolicited tasks. Additionally, since there is a lack of test environments for bots under development, bot developers are forced to *test bots in production.*

4.2.3   Different perceptions of noise. Bots' verbosity, high frequency of actions, and the execution of unsolicited tasks are generally *perceived as* **noise** by human developers. This perception, however, might be *influenced by* **project standards** (3) and **developers' previous experiences** (3), as noted by P12 "*what some people might think of as noise is information to other people, right? Like, it depends on the user's role and context within the project.*" In some cases, for example, an experienced developer may be annoyed by a large amount of information, while a newcomer may benefit from it. As explained by P3, an experienced open source maintainer, "*when it is your self maintained project, and you see these comments everywhere and you cannot configure the [bot] to turn it off, it might become just noise.*" For P19, a verbose bot is "*really more for novices*" since it "*tends to have pretty, pretty dense messages.*" However, dense messages are not necessarily useful for a developer, nor will a new contributor necessarily benefit from them. For P7, newcomers could perceive the bots' verbosity as noise: "*I do worry that newcomers perceive the bots as noisy, even with only 1 or 2 comments, because the comments are large.*" In addition, maintainers claim that bots' behaviors might be perceived as noisy when they do not comply with projects' rules and standards. P9 provided an example of this: "*every public repository has some standards, whether in terms of communication, whether in terms of how many messages the developer should see. And the bot likely will not comply with this policy.*"

4.2.4   Bot overpopulation. In addition to project standards and developers' previous experience, **bot overpopulation** (8) might also influence the perception of noise. Eight interviewees reported that annoying bot behaviors can be *intensified* by the presence of several bots on the same repository. As said by P19: "*because there were 30 different bots, and each one of them was asynchronously going in. So, it was just giving us tons and tons of comments.*"

4.2.5   Effects of information overload. The bots' annoying behaviors, which are perceived as noise, lead to **information overload** (7). As stated by P3: "*it [bot comment] replicates information that we already had.*" Also, the overload of information can be seen as an *overload of notifications* (e.g.,

emails or GitHub notifications). It is a problem, as explained by P12: "*given that we already have a lot of notifications for those of us who use GitHub a lot, then I think that's a real problem.*"

Therefore, the information overload negatively impacts both **human communication** (6) and **development workflow** (5). Developers mentioned that bots *interrupt the conversation flow* in pull requests, adding other information in the middle of the conversation: "*you are talking to the person who submitted the pull request and then a bot comes in and puts other information in the middle of your conversation*" [P13]. Participants also mentioned that they usually "*miss important comments from humans*" [P1] among the avalanche of information. Due to information overload, it is also hard to *parse all the data* to extract something meaningful. Project maintainers often complain about being interrupted by bot notifications, which disrupts the development workflow. They also started to deal with the *burden of checking* whether it is a human or bot notification. Their time and efforts are also consumed by other tasks not related to development, including reporting spam and excluding undesirable bot comments: "*I waste five minutes determining that it is a spam*" [P5].

One practical example of the effects of noise introduction is the case of *mention bot*. The challenges of using this bot were reported in the literature by Peng and Ma [43], Peng et al. [44] and mentioned by P5. Mention bot is a reviewer recommendation bot created by Facebook. The main role of this bot is to suggest to a reviewer a specific pull request. In a project that P5 helps to maintain, a maintainer that no longer works on the project started to receive several notifications when the bot was installed.

4.2.6 Countermeasures to overcome noise. We also grouped the strategies that our participants recommend to overcome bots' noise. In most cases, participants reported the **countermeasures** (6) as a way to manage the noise rather than avoid it. For instance, the noise continues to happen even when a developer *stops watching a repository*. Maintainers also mentioned that they need to *re-configure* the bot to avoid some behaviors. For some bots, it is possible to turn the comments off. During member-checking, P20 reported that in some cases it necessary to *re-configure the bot* to "*lower the frequency of actions.*" For example, this is useful for reducing the overload of information generated by dependency management bots, which can submit a couple of pull requests every day. These bots can suddenly monopolize the continuous integration and disrupt the workflow for humans. Another countermeasure that emerged from member-checking is the necessity to *re-design* the bot. P12 mentioned that, after receiving feedback from contributors about noise, they decided to re-design the content of the bot messages and when the bot would be allowed to interact on pull requests.

> **Summary of the noise theory.** We presented a theory of how certain bot behaviors can be perceived as noise on OSS pull requests. This perception often relies on the number of bots on a repository, project standards, and the human's previous experience. In short, we found that the noise introduced by bots leads to information overload, which interferes with how humans communicate, work, and collaborate on social coding platforms.

5 DISCUSSION

In this section, we discuss our main findings, comparing them with the state-of-the-art. Further, we discuss the implications of this work for the OSS community, bot developers, social coding platforms, and researchers.

Bots on GitHub serve as an interface to integrate humans and services [53, 61]. They are commonly integrated into the pull request workflow to automate tasks and communicate with human

developers. The increasing number of bots on GitHub relates to the growing importance of automating activities around pull requests. However, as discussed by Storey and Zagalsky [53] and Paikari and van der Hoek [41], potentially negative impacts of task automation through bots are overlooked. Therefore, it is critical to understand software bots as socio-technical—rather than technical—applications, which must be designed to consider human interaction, developers' collaboration, and other ethical concerns [52]. In this context, our work contributes by introducing and systematizing evidence from the perspective of OSS practitioners who have experience interacting with and developing bots on the GitHub platform.

Bot communication challenges. The way bots communicate impacts developers' interpretations and how they handle bot outcomes. According to Lebeuf et al. [27], the way bots communicate is important because "*the bot's purpose – what it can and can't do – must be evident and match user expectations.*" However, we evidenced the necessity of *previous technical knowledge to interact* with and understand the messages of bots on GitHub. Combined with the *lack of context*, it might be extremely difficult for humans to extract meaningful guidance from bots' feedback. These challenges relate to the *platform limitations* bot developers face and the textual communication channel [29]. These findings complement the previous literature, which found that practitioners often complain that bots have poor communication skills and do not provide feedback that supports developers' decisions [61]. Brown and Parnin [3] argue that designing bots to provide actionable feedback for developers is still an open challenge.

Expectations Breakdowns. Developers with different profiles and backgrounds have different expectations about bot interaction. Bots, for example, *enforce predefined cultural rules* of a community, causing expectation breakdowns for outsiders. We also found that *bots intimidate newcomers*. New contributors might be confused when interacting with a bot that they have never seen or heard of before. Previous work by Wessel et al. [61] has already mentioned that support for newcomers is both challenging and desirable. In a subsequent study, Wessel et al. [62] reported that although bots could make it easier for some newcomers to submit a high-quality pull request, bots can also provide them with information that can lead to rework, discussion, and ultimately dropping out from contributing. Developers' different cognitive styles [31, 60] may also have diverse expectations and their profiles should be considered during the design of bot messages to avoid expectation breakdowns. Differences related to developers' backgrounds are a common cause of problems in distributed software development [48]. However, when it comes to bots interacting on social coding platforms, it is still an under-explored theme.

Ethical challenges. *Intrusive bots* generate ethical concerns. Common intrusive bot behaviors include modifying actions performed by humans, such as changing commits or pull requests content, or even spamming repositories with unsolicited pull requests or comments. Spamming by bots is one of the factors responsible for the perception of noise on GitHub repositories. Another important concern is whether bots are allowed to impersonate humans [52]. For bots on Wikipedia, for example, this behavior is expressly prohibited [37]. At the same time, Murgia et al. [38] have shown that individuals on Stack Overflow might be more likely to accept bots impersonating humans as opposed to bots disclosing that they are bots. On GitHub, however, there is no explicit prohibition for *bots impersonating humans*, or even *bots with malicious intent*. Thus, these bots might reinforce stereotypes and toxic behaviors, appear insincere, and target minorities. Golzadeh et al. [16] propose a strategy to detect bots on GitHub based on their message patterns. This strategy might be used to identify malicious bots.

Noise as a central challenge. Noise is a central challenge in bots' interactions on OSS' pull requests. We organized our findings into a theory that provides a broader vision of how certain

bot behaviors can be perceived as noise, how this impacts developers, and how they have been attempting to handle it. In communication studies and information theory, the term "noise" refers to anything that interferes with the communication process between a speaker and an audience [46]. In the context of social coding platforms, we found that the noise introduced by bots around pull requests refers to any interference produced by a bot's behavior that disrupts the communication between project maintainers and contributors.

Although we considered annoying bot behaviors as a source of noise, the perception of such noise varies. Although the overuse of bots potentializes the noise, as also noticed by Erlenhov et al. [10], we found that noise perception also depends on the experience and preferences of the developer interacting with the bot. For example, while a new contributor may benefit from receiving one or more detailed bot comments with guidance or feedback, an experienced maintainer may feel frustrated and annoyed by seeing and receiving frequent notifications from those verbose comments. Furthermore, noise perception also relies on the differences in the developers' cognitive style and on the limitations humans face to cope with information. For example, Information Processing Theory, proposed by Miller [33] in the field of cognitive psychology, describes the limited capacity of humans to store current information in memory. Individuals will invest only a certain level of cognitive effort toward processing a set of incoming information.

A main complaint about noise from developers is the notification overload from bots interrupting the development workflow. Other studies focusing on a single bot also reported that developers can be overwhelmed by bot notifications [3, 34, 43, 44]. According to Erlenhov et al. [10], there is a trade-off between timely bot notifications and frequent interruptions and information overload. Our findings provide further detail on how developers deal with those notifications and the impacts on the development workflow. Developers deemed notification overload as a significant problem, since they already receive a large number of daily notifications. On GitHub specifically, Goyal et al. [19] found that active developers typically receive dozens of public event notifications each day, and a single active project can produce over 100 notifications per day. The CSCW community for decades has been investigating awareness mechanisms based on notifications [30, 47], which have not been yet explored by social coding platforms. As pointed out by Iqbal and Horvitz [22], users want to be notified, but they also want to have ways to filter notifications and determine how they will be notified. Steinmacher et al. [48] has performed a systematic literature review on awareness support in distributed software development, which can be used to inspire the design of appropriate awareness mechanisms for social coding platforms.

Our interviewees mentioned the direct effects of information overload on their communication, including difficultly in managing the incoming information and the interruption in the flow of the conversation, which might incur the loss of important information. These effects of information overload have been already observed in teams that collaborate and communicate online [2, 23, 40]. According to Nematzadeh et al. [40], both the structure and textual contents of human conversation may be affected by a high information load, potentially limiting the overall production of new information in group conversations. In the context of our study, this change in the conversational dynamics described by Nematzadeh et al. [40] can impact the overall engagement of contributors and maintainers when discussing pull requests. Further, Jones et al. [23] proposed a theoretical model on the impact on message dynamics of individual strategies to cope with the information overload. According to Jones et al. [23], as the information overload grows, users tend to focus on and respond to simpler information, and eventually cease active participation.

5.1 Implications

The results of our study can help the software bot community improve the design of bots on ethical and interaction levels. In the following, we discuss how our results lead to practical implications for practitioners as well as insights and suggestions for researchers.

Implications for Bot developers: On the path toward making bots more effective for communicating with and helping developers, many design problems need to be solved. Any developer who wants to build a bot for integration into a social coding platform first needs to consider the impact that the bot may have on both technical and social contexts. Based on our results, further bot improvements can be envisioned. One of the biggest complaints about bot interaction is the repetitive actions they perform. In this way, to prevent bots from introducing communication noise, bot developers should know when and to what extent the bot should interrupt a human [29, 52]. In addition, bot developers should provide mechanisms to enable better configurable control over bot actions, rather than just turn off bot comments. Further, these mechanisms need to be explicitly announced during bot adoption (e.g., noiseless configuration, preset levels of information). Another important aspect of bot interaction is the way bots should display information to the developer. Developers often complain about bots providing verbose feedback (in a comment) instead of just status information. Therefore, bot developers also should identify the best way to convey the information (e.g., via status information, comments).

Another point to be considered is that bots spamming repositories was one of the most mentioned ethical challenges by OSS maintainers. It is important for bot developers to design an opt-in bot and provide maintainers control over bot actions. In addition, our study results underscored that some developers feel uncomfortable interacting with a bot. Human users can hold higher expectation with overly humanized bots (e.g., bots that say "*thank you*"), which can lead to frustration [15].

Implications for Social Coding Platforms: Because of the growing use of bots for collaborative development activities [9], a proliferation of bots to automate software development tasks was expected. Recently, GitHub introduced GitHub Actions[5], a feature providing automated workflows. These actions allow the automation of tasks based on various triggers and can be easily shared from one repository to another. However, the way these actions communicate in the GitHub platform is the same as bots [25], which can lead to the same interaction challenges presented in this study.

Our findings also reveal that there are some limitations imposed by the GitHub platform that restrict the design of bots. In short, the platform restrictions might limit both the extent of bot actions and the way bots communicate. It is essential to provide a more flexible way for bots to interact, incorporating rich user interface elements to better engage users. At the same time, there is a need for well-defined governance roles for bots on GitHub, as already established by Wikipedia [37]. Therefore, it is important that bots have a documentation page that clearly describes their propose and what they can do on each repository. Also, it is important to have easy mechanisms so project maintainers can turn off or pause a bot at any time.

Based on the premise that users would like to have better control over their notifications [22], GitHub should also provide a mechanism to filter out real notifications from bot ones. This would facilitate the management of bot notifications and avoid wasting developers' time filtering non-humans content. Further, the detection of non-human notifications would help developers identify pull requests that are merely spam.

Implications for Researchers: We identified a set of 25 challenges to developing, adopting, and interacting with bots on social coding platforms. Part of these challenges can be addressed by

---

[5]https://github.com/features/actions

leveraging machine learning techniques to enrich bots. Thus, we believe that there is an opportunity for future research to support OSS projects by developing smarter bots, thereby providing better human-bot communication. For example, bots could understand the context of their actions and provide actionable changes or suggestions for developers. To design effective bots to support developers on OSS projects, there is room for research on how to combine the knowledge on building bots and modeling interactions from other domains with the techniques and approaches available in software engineering.

Considering that bot output is mostly text-based, how bots present the content can highly impact developers' perceptions [29]. Still, as aforementioned, the developers' cognitive styles might influence how developers interpret the bot comments' content. In this way, future research can investigate how people with different cognitive styles handle bot messages and learn from them. Future research can lead to a set of guidelines on how to design effective messages for different cognitive styles and developer profiles. Further, it is also important to understand how the content of bot messages influences developers' emotions. To do so, researchers can analyze how developers' emotions expressed in comments changed following bot adoption.

Another challenge is related to the information overload caused by bot behavior on pull requests, which has received some attention from the research community [10, 61, 64], but remains a challenging problem. In fact, there is room for improvement on human-bot collaboration on social coding platforms. Possible future research can leverage noise theory to better support bots' social interaction in the context of OSS. In addition, when they are overloaded with information, teams must adapt and change their communication behavior [8]. Therefore, there is also an opportunity to investigate changes in developers' behavior imposed by the effects of information overload.

6 LIMITATIONS AND THREATS TO VALIDITY

As any empirical research, our research presents some limitations and potential threats to validity. In this section, we discuss them, their potential impact on the results, and how we have mitigated these limitations.

Scope of the results: Our findings are grounded in the qualitative analysis of data from practitioners who are experienced with bots on the GitHub platform. Hence, our theory of noise introduced by bots may not necessarily represent the context of other social coding platforms, such as GitLab and Bitbucket.

Data representativeness: Although we interviewed a substantial number of developers, we likely did not discover all possible challenges or provide full explanations of the challenges. We are aware that each project has its singularities and that the OSS universe is huge, meaning the bots' usage and the challenges incurred by those bots can differ according to the project or ecosystem. Our strategy to consider different developer profiles aimed to alleviate this threat, identifying recurrent mentions of challenges from multiple perspectives. Our interviewees were also diverse in terms of the number of years of experience with software development and bots.

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 [54], sampling should 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 bot developers and developers who are contributors and/or maintainers of OSS projects. Although we interviewed only 3 contributor-only

developers, the analysis of their interviews did not provide new insights when compared to the maintainers who were also contributors.

Reliability of results: To increase the construct validity and improve the reliability of our findings, we employed a constant comparison method [14]. In this method, each interpretation is constantly compared with existing findings as it emerges from the qualitative analysis. In addition, we also conducted member-checking. During member-checking, participants confirmed our interpretation of the results, requesting only minor changes.

7 CONCLUSION

The literature on bots on social coding platforms report several potential benefits, such as reducing maintainers' effort on repetitive tasks [62] and increasing productivity [10]. In this paper, we investigated the challenges of using bots to support pull requests. We conducted 21 semi-structured interviews with open source developers experienced with bots. We found several challenges regarding the development, adoption, and interaction of bots on pull requests of OSS projects.

Among the existing challenges, the introduction of noise is the most pressing one. Developers frequently complained about annoying bot behaviors on pull requests, which can be perceived as noise. Noise leads to information overload, which disrupts both human communication and development workflow. Towards managing the noise effects, project maintainers often take some countermeasures, including re-designing the bot's interaction, re-configuring the bot, and not watching a repository. Compared to the previous literature, our findings provide a comprehensive understanding of the interaction problems caused by the use of bots in pull requests.

Our study opens the door for researchers and practitioners to further understand the challenges introduced by adopted bots to save developers time and efforts on social coding platforms. For future work, we plan to design and evaluate strategies to mitigate problems related to information overload incurred by the interaction of developers with software bots on social coding platforms, thereby assisting developers to communicate and accomplish their tasks.

ACKNOWLEDGMENTS

REFERENCES

[1] Norah Abokhodair, Daisy Yoo, and David W. McDonald. 2015. Dissecting a Social Botnet: Growth, Content and Influence in Twitter. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. ACM, New York, NY, USA, 839–851. https://doi.org/10.1145/2675133.2675208

[2] David Bawden and Lyn Robinson. 2009. The dark side of information: overload, anxiety and other paradoxes and pathologies. *Journal of information science* 35, 2 (2009), 180–191.

[3] 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)*.

[4] 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 *Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 161–171.

[5] Kathy Charmaz. 2006. *Constructing grounded theory: A practical guide through qualitative analysis.* sage.

[6] Dan Cosley, Dan Frankowski, Loren Terveen, and John Riedl. 2007. SuggestBot: Using Intelligent Task Routing to Help People Find Work in Wikipedia. In *Proceedings of the 12th International Conference on Intelligent User Interfaces* (Honolulu, Hawaii, USA) *(IUI '07)*. ACM, New York, NY, USA, 32–41. https://doi.org/10.1145/1216295.1216309

[7] Robert Dale. 2016. The return of the chatbots. *Natural Language Engineering* 22, 5 (2016), 811–817.

[8] Thomas Ellwart, Christian Happ, Andrea Gurtner, and Oliver Rack. 2015. Managing information overload in virtual teams: Effects of a structured online team adaptation on cognition and performance. *European Journal of Work and Organizational Psychology* 24, 5 (2015), 812–826.

[9] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Riccardo Scandariato, and Philipp Leitner. 2019. Current and Future Bots in Software Development. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 7–11. https://doi.org/10.1109/BotSE.2019.00009

[10] 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)*.

[11] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. *Commun. ACM* 59, 7 (2016), 96–104.

[12] R. Stuart Geiger and Aaron Halfaker. 2017. Operationalizing Conflict and Cooperation Between Automated Software Agents in Wikipedia: A Replication and Expansion of 'Even Good Bots Fight'. *Proceedings of the ACM Human-Computer Interaction* 1, CSCW, Article 49 (Dec. 2017), 33 pages. https://doi.org/10.1145/3134684

[13] Supratip Ghose and Jagat Joyti Barua. 2013. Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor. In *Proceedings of the 2013 International Conference on Informatics, Electronics & Vision (ICIEV)*. IEEE, Washington, DC,USA, 1–5.

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

[15] Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. 2017. Towards Designing Cooperative and Social Conversational Agents for Customer Service. In *Proceedings of the International Conference on Information Systems (ICIS)*. AIS.

[16] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. 2021. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software* 175 (2021), 110911. https://doi.org/10.1016/j.jss.2021.110911

[17] Georgios Gousios, Margaret-Anne Storey, and Alberto Bacchelli. 2016. Work Practices and Challenges in Pull-based Development: The Contributor's Perspective. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. ACM, New York, NY, USA, 285–296. https://doi.org/10.1145/2884781.2884826

[18] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie van Deursen. 2015. Work Practices and Challenges in Pull-based Development: The Integrator's Perspective. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1* (Florence, Italy) *(ICSE '15)*. IEEE Press, Piscataway, NJ, USA, 358–368. http://dl.acm.org/citation.cfm?id=2818754.2818800

[19] Raman Goyal, Gabriel Ferreira, Christian Kästner, and James Herbsleb. 2018. Identifying unusual commits on GitHub. *Journal of Software: Evolution and Process* 30, 1 (2018), e1893.

[20] Tim Healy. 2012. The unanticipated consequences of technology. *Nanotechnology: ethical and social Implications* (2012), 155–173.

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

[22] Shamsi T Iqbal and Eric Horvitz. 2010. Notifications and awareness: a field study of alert usage and preferences. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. 27–30.

[23] Quentin Jones, Gilad Ravid, and Sheizaf Rafaeli. 2004. Information overload and the message dynamics of online interaction spaces: A theoretical model and empirical exploration. *Information systems research* 15, 2 (2004), 194–210.

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

[25] Timothy Kinsman, Mairieli Wessel, Marco Gerosa, and Christoph Treude. 2021. How Do Software Developers Use GitHub Actions to Automate Their Workflows?. In *Proceedings of the IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE.

[26] 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 *Proceedings of the International Conference on Fuzzy Systems*. IEEE, Washington, DC, USA, 1–8.

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

[28] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why developers are slacking off: Understanding how software teams use slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*. ACM, 333–336.

[29] Dongyu Liu, Micah J. Smith, and Kalyan Veeramachaneni. 2020. Understanding User-Bot Interactions for Small-Scale Automation in Open-Source Development. In *Proceedings of the 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

[30] Gustavo López and Luis A Guerrero. 2016. Ubiquitous notification mechanism to provide user awareness. In *Advances in Ergonomics in Design.* Springer, 689–700.

[31] C. Mendez, H. S. Padala, Z. Steine-Hanson, C. Hildebrand, A. Horvath, C. Hill, L. Simpson, N. Patil, A. Sarma, and M. Burnett. 2018. Open Source Barriers to Entry, Revisited: A Sociotechnical Perspective. In *Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. 1004–1015.

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

[33] 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.

[34] 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

[35] 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

[36] KF Mulder. 2013. Impact of new technologies: how to assess the intended and unintended effects of new technologies. *Handb. Sustain. Eng.(2013)* (2013).

[37] Claudia Müller-Birn, Leonhard Dobusch, and James D Herbsleb. 2013. Work-to-rule: the emergence of algorithmic governance in Wikipedia. In *6th International Conference on Communities and Technologies.* ACM, 80–89.

[38] 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.* 1272–1279.

[39] Kazuaki Nakamura, Koh Kakusho, Tetsuo Shoji, and Michihiko Minoh. 2012. Investigation of a Method to Esti-mate 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.

[40] 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).

[41] Elahe Paikari and André van der Hoek. 2018. A Framework for Understanding Chatbots and Their Future. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering* (Gothenburg, Sweden) *(CHASE '18)*. ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/3195836.3195859

[42] Michael Quinn Patton. 2014. *Qualitative research & evaluation methods: Integrating theory and practice.* Sage publica-tions.

[43] 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

[44] 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* (Montreal, QC, Canada) *(ChineseCHI '18)*. ACM, New York, NY, USA, 152–155. https://doi.org/10.1145/3202667.3202694

[45] Saiph Savage, Andres Monroy-Hernandez, and Tobias Höllerer. 2016. Botivist: Calling volunteers to action using online bots. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing.* ACM, New York, NY, USA, 813–822.

[46] C. E. Shannon. 2001. A Mathematical Theory of Communication. *SIGMOBILE Mob. Comput. Commun. Rev.* 5, 1 (Jan. 2001), 3–55. https://doi.org/10.1145/584091.584093

[47] Carla Simone, Monica Divitini, and Kjeld Schmidt. 1995. A notation for malleable and interoperable coordination mechanisms for CSCW systems. In *Proceedings of conference on Organizational computing systems.* 44–54.

[48] Igor Steinmacher, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Awareness support in distributed software development: A systematic review and mapping of the literature. *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW)* 22, 2-3 (2013), 113–158.

[49] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. ACM, New York, NY, USA, 1379–1392. https://doi.org/10.1145/2675133.2675215

[50] 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).

[51] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded theory in software engineering research: a critical review and guidelines. In *Proceedings of the 38th International Conference on Software Engineering.* 120–131.

[52] 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.

[53] 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

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

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

[56] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Let's Talk About It: Evaluating Contributions Through Discussion in GitHub. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Hong Kong, China) *(FSE 2014).* ACM, New York, NY, USA, 144–154. https://doi.org/10.1145/2635868.2635882

[57] 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

[58] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015).* Association for Computing Machinery, New York, NY, USA, 805–816. https://doi.org/10.1145/2786805.2786850

[59] Alessandro Vinciarelli, Anna Esposito, Elisabeth André, Francesca Bonin, Mohamed Chetouani, Jeffrey F Cohn, Marco Cristani, Ferdinand Fuhrmann, Elmer Gilmartin, Zakia Hammal, et al. 2015. Open challenges in modelling, analysis and synthesis of human behaviour in human–human and human–machine interactions. *Cognitive Computation* 7, 4 (2015), 397–413.

[60] 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

[61] 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

[62] Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco Aurelio Gerosa. 2020. What to Expect from Code Review Bots on GitHub? A Survey with OSS Maintainers. In *Proceedings of the SBES 2020 - Ideias Inovadoras e Resultados Emergentes.*

[63] 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 *Proceedings of the IEEE International Conference on Software Maintenance and Evolution.* IEEE Computer Society.

[64] Mairieli Wessel and Igor Steinmacher. 2020. The Inconvenient Side of Software Bots on Pull Requests. In *Proceedings of the 2nd International Workshop on Bots in Software Engineering (BotSE).* https://doi.org/10.1145/3387940.3391504

[65] David D Woods and Emily S Patterson. 2001. How unexpected events produce an escalation of cognitive and coordinative demands. *PA Hancock, & PA Desmond, Stress, workload, and fatigue. Mahwah, NJ: L. Erlbaum* (2001).

[66] 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

[67] Bin Xu, Tina Chien-Wen Yuan, Susan R. Fussell, and Dan Cosley. 2014. SoBot: Facilitating Conversation Using Social Media Data and a Social Agent. In *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Baltimore, Maryland, USA) *(CSCW Companion '14).* ACM, New York, NY, USA, 41–44. https://doi.org/10.1145/2556420.2556789

[68] Lei Zheng, Christopher M Albano, and Jeffrey V Nickerson. 2018. Steps toward Understanding the Design and Evaluation Spaces of Bot and Human Knowledge Production Systems. In *Proceedings of the Wiki Workshop'19.*

[69] Victor W Zue and James R Glass. 2000. Conversational interfaces: Advances and challenges. *Proc. IEEE* 88, 8 (2000), 1166–1180.