Perception of Software Bots on Pull Requests on Social Coding Environments

Mairieli Santos Wessel

Thesis presented to the
Institute of Mathematics and Statistics
of the University of São Paulo
in partial fulfillment
of the requirements
for the degree of
Doctor of Science

Program:	Computer Science
Advisor:	Prof. Dr. Marco Aurelio Gerosa

This research is supported by CNPq

São Paulo
August 16, 2021

Perception of Software Bots on Pull Requests on Social Coding Environments

Mairieli Santos Wessel

Resumo

Mairieli Santos Wessel. **Percepção sobre Software Bots em Pull Requests em Ambientes Sociais de Codificação**. Tese (Doutorado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

Software *bots* são aplicações que integram seu trabalho a tarefas humanas, servindo de interface entre os usuários e outras ferramentas. Devido à sua capacidade de automatizar tarefas, os *bots* se tornaram relevantes para projetos de software livre hospedados na plataforma GitHub. Geralmente, tais projetos usam *bots* para automatizar uma variedade de tarefas, por exemplo, garantir a assinatura do contrato de licença, relatar falhas de integração contínua, revisar o código e solicitar revisões, fazer a triagem de problemas e refatorar o código-fonte. No entanto, por meio de estudos preliminares evidenciamos que a interação desses *bots* em pull requests pode ser perturbadora para os contribuidores e mantenedores dos projetos. Embora os *bots* possam ser úteis para apoiar o trabalho dos mantenedores, às vezes seus comentários são considerados como *spams* e são rapidamente ignorados pelos contribuidores. Nesta tese, o objetivo foi identificar e compreender os desafios enfrentados pelos mantenedores e contribuidores durante a interação com *bots* em pull requests, e projetar e avaliar estratégias que ajudem a mitigar os desafios encontrados. Para esse fim, conduzimos um conjunto de estudos, utilizando diferentes métodos de pesquisa. Para identificar os desafios de interação com os *bots*, entrevistamos 21 profissionais, incluindo mantenedores e contribuidores de projetos de código aberto, e desenvolvedores de *bots*. Os dados foram analisados qualitativamente por meio de codificação aberta e axial. Tal análise resultou em uma teoria sobre como os desenvolvedores percebem os comportamentos irritantes dos *bots* como ruídos. Com base nessa teoria, conduzimos um estudo participativo empregando *design fiction* com 32 profissionais e pesquisadores. Para entender a percepção dos participantes sobre a solução idealizada, realizamos um estudo com um subconjunto contendo 15 participantes do *design fiction*. Após concluir essa etapa, identificamos um conjunto de melhorias para o protótipo de acordo com as sugestões recebidas dos participantes do estudo. As principais contribuições desta tese são: (i) elucidação das mudanças nos projetos após a adoção de um ou mais *bots*; (ii) uma teoria sobre como o ruído introduzido por *bots* atrapalha a comunicação e o fluxo de trabalho dos desenvolvedores em projetos de código aberto; (iii) um conjunto de estratégias para mitigar a sobrecarga de informações gerada pela interação dos *bots* em pull requests; e (iv) o conceito de um *meta-bot* para apoiar a contribuição para projetos de código aberto. Essas contribuições podem ajudar os profissionais a entender os efeitos da adoção de um *bot*, e os pesquisadores e designers de ferramentas podem utilizar nossos resultados para melhor apoiar a interação humano-bot em plataformas sociais de codificação.

Palavras-chave: Software Bots. GitHub Bots. Interação Humano-bot. Open Source Software. Desenvolvimento Colaborativo. Engenharia de Software.

Abstract

Mairieli Santos Wessel. **Perception of Software Bots on Pull Requests on Social Coding Environments**. Thesis (Doctorate). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2021.

Software bots integrate their work with humans' tasks, serving as conduits between users and other tools. Due to their ability to automate tasks, bots have become particularly relevant for Open Source Software (OSS) projects hosted on GitHub. Commonly, projects use bots to automate various tasks, such as ensuring license agreement signing, reporting continuous integration failures, reviewing code and pull requests, triaging issues, and refactoring the source code. However, in preliminary studies, our findings indicate that the interaction of these bots on pull requests can be disruptive and perceived as unwelcoming by contributors and maintainers. Although bots can be useful for supporting maintainers' work, sometimes their comments are seen as spam and are quickly ignored by contributors. In this dissertation, our goal was to identify and understand challenges maintainers and contributors face during interaction with bots on pull requests of OSS projects and design and evaluate a software bot that mitigates some of these problems. Toward this end, we conducted multiple studies using multiple research methods. To identify the challenges caused by bots in pull request interactions, we interviewed 21 practitioners, including project maintainers, contributors, and bot developers. The data was qualitatively analyzed using open and axial coding. Subsequently, the analysis resulted in a theory of how human developers perceive annoying bot behaviors as noise on social coding platforms. Based on this theory, we conducted a participatory design fiction study with 32 practitioners and researchers. This study resulted in design strategies that served as insights to create a prototype. We conducted a suitability study with 15 design fiction participants to assess the envisioned solution. By collecting participants' perceptions about a prototype implementing the envisioned strategies, we identified improvements to the prototype according to the suggestions received from the study participants. The main contributions of this dissertation are: (i) identifying the changes in project activity indicators after the adoption of a bot; (ii) proposing a theory about how noise introduced by bots disrupts developers' communication and workflow; (iii) identifying strategies to mitigate the information overload generated by the existing bots' interaction; and (iv) the concept of a meta-bot to support contribution to OSS projects. These contributions may help practitioners understand the effects of adopting a bot. Researchers and tool designers may leverage our results to better support human-bot interaction on social coding platforms.

Keywords: Software Bots. GitHub Bots. Human-bot Interaction. Open Source Software. Collaborative Development. Software Engineering.

List of Abbreviations

ACM	Association for Computing Machinery
AI	Artificial Intelligence
API	Application Programming Interfaces
CI	Continuous Integration
DF	Design Fiction
IEEE	Institute of Electrical and Electronics Engineers
IME	Instituto de Matemática e Estatística
ML	Machine Learning
MWW	Mann-Whitney-Wilcoxon
NPL	Natural Language Processing
OSS	Open Source Software
PR	Pull Request
Q&A	Questions and Answers
RDD	Regression Discontinuity Design
SE	Software Engineering
USP	Universidade de São Paulo

List of Figures

List of Tables

Contents

Chapter 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 (Tsay *et al.*, 2014). Pull requests facilitate interaction among developers to review and integrate code contributions. In the pull-based development model, project maintainers carefully inspect code changes and engage in discussion with contributors to understand and improve the modifications before integrating them into the codebase (McIntosh *et al.*, 2014). The time maintainers spend reviewing pull requests is non-negligible and can affect, for example, the volume of new contributions (Yu *et al.*, 2015) and the onboarding of newcomers (I. Steinmacher, I. Wiese, *et al.*, 2013).

Software bots play a prominent role in the pull request review process (Wessel, Souza, *et al.*, 2018). These automation tools serve as an interface between users and other tools (Storey and Zagalsky, 2016) and reduce the workload of maintainers and contributors. Accomplishing tasks that were previously performed solely by human developers, and interacting in the same communication channels as their human counterparts, bots have become new voices in the pull request conversation (Monperrus, 2019). Throughout comments on pull requests, bots guide contributors to provide necessary information before maintainers triage the pull requests (Wessel, Souza, *et al.*, 2018). To alleviate their workload (Gousios, Storey, *et al.*, 2016), 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 (Vasilescu *et al.*, 2015; D. Kavaler *et al.*, 2019). More complex tasks include repairing bugs (Urli *et al.*, 2018; Monperrus, 2019), refactoring source code (Wyrich and Bogner, 2019), recommending tools (Brown and Parnin, 2019), updating dependencies (Mirhosseini and Parnin, 2017), and fixing static analysis violations (Carvalho *et al.*, 2020).

The introduction of bots aims to save cost, effort, and time (Storey and Zagalsky, 2016), allowing maintainers to focus on development and review tasks. However, new technology often brings consequences that counter designers' and adopters' expectations (Healy, 2012). 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 (Woods and Patterson, 2001). For example, we have shown that although the number of human comments decreases after the introduction of bots (Wessel, Serebrenik, I. S. Wiese, *et al.*, 2020), many developers do not perceive this decrease (Wessel, Serebrenik, I. Wiese, I. Steinmacher, and Marco Aurelio Gerosa, 2020). 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 complaining on social media: "*Whoever wrote [bot-name] fundamentally does not understand software development.*"[1] Moreover, as bots have become new voices in developers' conversation (Monperrus, 2019), they may overburden developers who already suffer from information overload when communicating online (Nematzadeh *et al.*, 2016). On an abandoned pull request, a maintainer noted the frequency of actions of a bot: "*@<bot-name> seems pretty active here [...].*"[2] As the introduction of a technology may provoke changes in human behavior (Mulder, 2013), it is important to understand how bots affect the group dynamics; yet, this is often neglected (Storey and Zagalsky, 2016; Paikari and Hoek, 2018).

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 (2017), for example, report that maintainers are overwhelmed by bot pull request notifications, which interrupt their workflow. According to Brown and Parnin (2019), the human-bot interaction on pull requests can be inconvenient, leading developers to abandon their contributions due to poor bots' design. This problem may be especially relevant for newcomers, who require special support during the onboarding process due to the barriers they face (I. Steinmacher, T. Conte, *et al.*, 2015; I. Steinmacher, T. U. Conte, *et al.*, 2016). 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.

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 (Storey, Serebrenik, *et al.*, 2020; Liu *et al.*, 2020). Designers should envision software bots as socio-technical rather than purely technical applications, considering human interaction, developers' collaboration, and ethical concerns (Storey and Zagalsky, 2016). 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. Considering this context, in this dissertation we focus on the challenges incurred by the use of software bots on the pull requests' workflow. With a more in depth understanding of the challenges incurred by the bots' interaction, researchers and practitioners can invest their efforts in designing or improving bots, ultimately supporting developers on submitting and reviewing pull requests.

[1] https://twitter.com/mojavelinux/status/1125077242822836228

[2] https://github.com/facebook/react/pull/12457#issuecomment-413429168

1.1 Research Questions

The goal of this research is to identify and understand challenges introduced by the interaction of bots on pull requests of open-source projects, and design and evaluate design strategies to mitigate these problems. To achieve this goal, we defined three research questions:

RQ1. How do pull request activities change after a bot is adopted in a project?

Since bots may bring unexpected impacts to group dynamics, as frequently occurs with new technology adoption, understanding and anticipating such effects is important for planning and management. Our results indicate that the adoption of bots, in fact, changes the dynamics of pull request activities. These results motivated us to investigate further developers' perspectives on the overall effects of introducing bots to open-source projects.

RQ2. What interaction challenges do bots introduce when supporting pull requests?

To understand the impact of bots interaction in-depth, we then focused on investigating the challenges incurred by bots interaction on pull requests. The main contribution for this research question is a theory of how human developers perceive annoying bot behaviors as noise on social coding platforms.

RQ3. What design strategies can potentially reduce the noise created by bots on pull requests?

As noise emerged as a central challenge in our analysis, we further investigated how to overcome it. This analysis resulted in a set of design strategies to enhance both bots and the GitHub platform. Our main findings indicate that a meta-bot mediator of other bots is a promising approach to handle the information overload from existing bots.

1.2 Research Design and Dissertation Organization

In this dissertation, we use multiple empirical methods to answer the three research questions. More specifically, we follow a mixed-methods approach with a sequential explanatory strategy, combining data analysis of GitHub data with semi-structured interviews conducted with open source developers, qualitative survey analysis, participatory design fiction, and other experimental studies. The research design comprises three phases and several complementary studies, as presented in Figure 1.1.

Warm-up – Characterization of GitHub bots. This phase consists of studies conducted during the definition of this dissertation' scope and helped us to define our research motivation. In the first study (S1), we conducted a preliminary study to characterize the

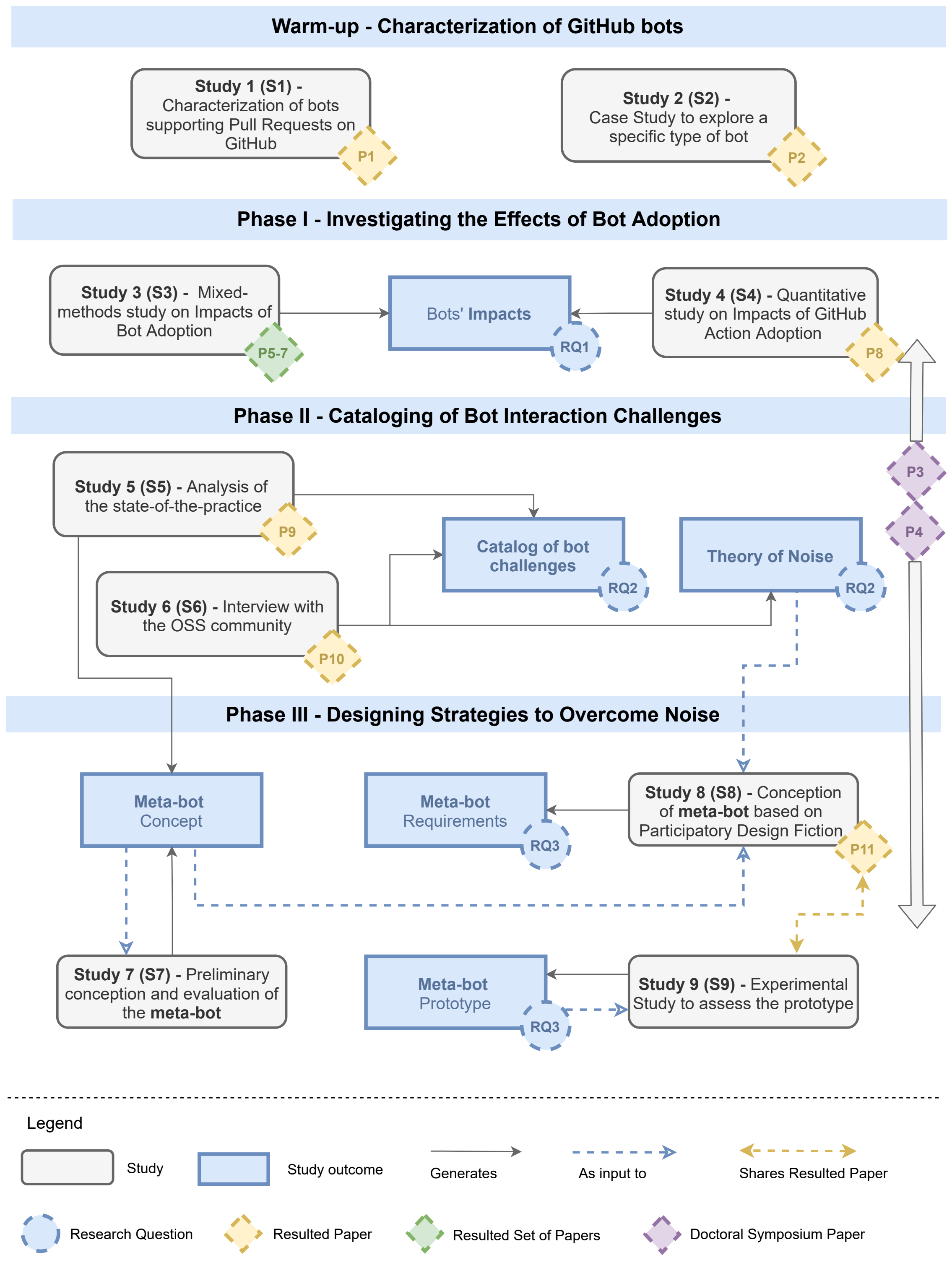

Warm-up - Characterization of GitHub bots
Study 1 (S1) - Characterization of bots supporting Pull Requests on GitHub
P1
Study 2 (S2) - Case Study to explore a specific type of bot
P2
Phase I - Investigating the Effects of Bot Adoption
Study 3 (S3) - Mixed-methods study on Impacts of Bot Adoption
P5-7
Bots' Impacts
RQ1
Study 4 (S4) - Quantitative study on Impacts of GitHub Action Adoption
P8
Phase II - Cataloging of Bot Interaction Challenges
P3
P4
Study 5 (S5) - Analysis of the state-of-the-practice
P9
Catalog of bot challenges
RQ2
Theory of Noise
RQ2
Study 6 (S6) - Interview with the OSS community
P10
Phase III - Designing Strategies to Overcome Noise
Meta-bot Concept
Meta-bot Requirements
RQ3
Study 8 (S8) - Conception of meta-bot based on Participatory Design Fiction
P11
Study 7 (S7) - Preliminary conception and evaluation of the meta-bot
Meta-bot Prototype
RQ3
Study 9 (S9) - Experimental Study to assess the prototype
Legend
Study
Study outcome
Generates
As input to
Shares Resulted Paper
Research Question
Resulted Paper
Resulted Set of Papers
Doctoral Symposium Paper

Figure 1.1: *Overview of the Research Design*

bots that support pull requests on GitHub. Our results indicate that bots' adoption is indeed widespread in OSS projects hosted on GitHub. Bots perform several tasks, including ensuring license agreement signing, reporting continuous integration failures, reviewing code and pull requests, triaging issues, and refactoring source code (Wessel, Souza, *et al.*, 2018). We also openly asked contributors and maintainers about the "challenges of using bots" on pull requests. Several contributors complained about the way the bots interact, saying, for example, that the *bots provide non-comprehensive or poor feedback*. Contributors also complained that *bots introduce communication noise* and that there is a *lack of information on how to interact with the bot*. The contributors deemed the current bots as *not smart enough* and provided insights into the bots' potential new features, such as *improving notification and awareness*, *enhancing user interaction*, *improving communicability*, and *answering specific questions*. Our results suggest that *GitHub bots* serve as a useful way to access services and automate tasks; however, in terms of supporting developers' interaction, they are not as evolved as in other domains (e.g., education, customer service). These limitations may have influenced developers' perceptions when they reported that bots should be smarter and have better ways to interact.

After this first study, we conducted a case study to explore a specific bot, called *stale bot* (S2). This GitHub bot helps maintainers by automatically labeling and closing abandoned issues and pull requests. We looked into the bots' configuration settings defined by projects, and how these projects adapt and maintain the bot over time. Our results indicate that the configuration of the stale bot does not require too much effort from project maintainers (Wessel, I. Steinmacher, *et al.*, 2019). Most projects made no more than three changes in the configuration file since in approximately 83% of the projects the configuration file was modified three times or less. According to our analysis results, stale bot is a recommended solution to help maintainers triaging issues and pull requests that are not affecting the project and the developers, since the stale characteristics can be adapted for each project.

Phase I – Investigating the Effects of Bot Adoption. To further understand the effects of bot adoption, we focused on one of the most common types of bots we found in Study 1: code review bots. To understand what happens after the adoption of a bot, we used a mixed-methods approach (Easterbrook *et al.*, 2008) with a sequential explanatory strategy (Creswell, 2003) (S3), combining data analysis of GitHub data with semi-structured interviews conducted with open-source developers. We used a *Regression Discontinuity Design* (RDD) (Thistlethwaite and D. T. Campbell, 1960) to model the effects of code review bot adoption across 1,194 OSS projects hosted on GitHub. Afterward, to further shed light on our results, we conducted semi-structured interviews with practitioners, including open-source project maintainers and contributors experienced with code review bots. Our results indicate that the adoption of code review bots increases the number of monthly merged pull requests, decreases monthly non-merged pull requests, and decreases communication among developers. From the developers' perspective, these effects are explained by the transparency and confidence the bot comments introduce, in addition to the changes in the discussion focused on pull requests.

In a follow-up study (S4), we investigated whether the aforementioned project activity indicators change after adoption of the newest type of bot automation introduced by GitHub: GitHub Actions. We used a RDD to model the effect of Action adoption across 926 projects that had adopted GitHub Actions for at least 6 months. Our findings indicate that the adoption of GitHub Actions increases the number of monthly rejected pull requests and decreases the monthly number of commits on merged pull requests. This differs from code review bots effects, which might be explained by the variety of tasks performed by the GitHub Actions in the study, and consequently their impacts on pull request activities. Based on the results of Phase I, we confirm that bots change the pull request dynamics and found preliminary evidence of challenges. Thus, we broadly delved into the challenges of interacting with bots in Phase II. The outcomes of Phase I are summarized in Chapter 3.

Phase II — Cataloging Bot Interaction Challenges. This phase comprises two studies to identify interaction challenges introduced by bots around pull requests, aiming to answer RQ2. To promote this investigation, we looked for challenges reported by practitioners in the software repositories (S5). We manually analyzed pull requests, looking for (i) human users mentioning bots, and (ii) bots' interactions—such as opening, merging, or commenting on pull requests. We noticed that the bots used in pull requests indeed (i) overwhelm developers' communication with notifications and feedback, (ii) perform wrong actions, and (iii) are misused due to their poor documentation (WESSEL and I. STEINMACHER, 2020).

In another study (S6), 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. Since noise emerged as a central theme in our analysis, we further theorized about it, grounded in the data we collected (WESSEL, I. WIESE, *et al.*, 2021). The noise theory from S6 was used as the input for the following phase of this dissertation. We summarize the outcomes of this phase in Chapter 4.

Phase III — Designing Strategies to Overcome Noise. After identifying the human-bot interaction problems, we designed strategies to better support developers' work on pull requests. Researchers have proposed the use of a meta-bot to integrate and moderate the interactions of multiple bots. SADEDDIN *et al.* (2007) 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 Marco Aurelio GEROSA (2018) 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 (DAGLI, 2019) 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.* Compared to other GitHub bots, the meta-bot could provide additional value to the interaction of already existing bots through the key feature of summarizing other bots' outcomes to avoid information overload. First, we implemented the meta-bot and conducted a preliminary study to validate its concept (S7).

After this preliminary study, we decided to focus on involving the users during the design process, to anticipate issues and capture their needs and expectations beforehand. Therefore, we applied Design Fiction (Blythe, 2014) as a participatory method (S8) to explore strategies to overcome the information overload evidenced in Phase II. Design Fiction is frequently used in the Human-Computer Interaction (HCI) field to gain insights to create new or refined objects and stories for further discussion. By capturing the expectations of developers who interact with bots, we elicited design strategies to the creation of a meta-bot. Using Design Fiction, we presented to participants a fictional history of a meta-bot capable of better supporting developers' interactions on pull requests, operating as a mediator between developers and the existing bots. Participants answered questions to complete the end of the fictional story, raising concerns around the use of bots and the requirements of the meta-bot.

Then, 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 (S9) aiming to answer RQ3. 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. In Chapter 5, we summarized the outcomes of Phase III.

This dissertation is organized as following: in Chapter 2, we provide an outline of software bots concepts, followed by an overview of the state-of-the-art of bots in software engineering; in Chapter3, we describe the Phase I, which resulted in the preliminary findings that grounded our work; in Chapter 4, we detail the Phase II, which resulted in the noise theory; and, in Chapter 5, we report the research phase III, including details of the method we conducted to design and evaluate the prototype. Finally, in Chapter 6, we present a discussion about our work and future directions.

1.3 Claimed Contributions

We claim that this dissertation contains four main novel contributions, which map onto the research questions presented in Section 1.1, and are related to the artifacts generated in phases I, II, and III, as presented in Section . The aforementioned contributions are:

- **Empirical identification of the impacts of bots usage to OSS projects.** As a result of Phase I, our work contributes to the state-of-the-art by (i) empirically identifying the changes in project activity indicators after the adoption of a bot; and (ii) elucidating the open-source developers' perspective on the impacts of bots. Practitioners and maintainers may leverage our results to understand, or even predict, bot effects on their projects.

- **Empirical cataloging of challenges incurred by using bots to support contributions on OSS projects.** Phase II 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 scarce and diffuse challenges, reported as secondary results. These findings may guide developers to consider the implications of new bots as they design them.

- **Empirical identification of design strategies to future bots and their environment.** In addition to the understanding of challenges introduced by bots on pull requests, in Phase III we identified design strategies that could help to overcome information overload. To do so, we considered the expectations of maintainers, contributors, bot developers, and experienced researchers. Researchers and tool designers may also leverage our results to enhance bots' communication design, thereby better supporting human-bot interaction on social coding platforms.
- **A meta-bot to support contribution to OSS projects.** We applied some of the proposed bot design strategies to a practical setting by building a meta-bot prototype to support contributions to OSS projects on GitHub.

1.4 Other Results

This doctoral research resulted in scientific publications, undergraduate diploma thesis, open source artifacts, and participation in events. In the following subsections, we present these results.

1.4.1 Published Papers

During the PhD program, we published several papers related to this dissertation' topic. So far, the main results were published in papers at CSCW 2018, BotSE 2019, BotSE 2020, SBES - IIER 2020, ICSME 2020, MSR 2021, CSCW 2021, and a journal was submitted to the Special Issue of ICSME 2020 of the Empirical Software Engineering (EMSE). The summarized references for the papers originated from this research are:

Papers related to warm-up – Characterization of GitHub Bots

Paper 1. WESSEL, Mairieli; DE SOUZA, Bruno M.; STEINMACHER, Igor; WIESE, Igor Scaliante; POLATO, Ivanilton; CHAVES, Ana Paula; GEROSA, Marco Aurélio. *The Power of Bots: Characterizing and Understanding Bots in Open Source Software Projects.* In: **ACM Conference on Computer Supported Cooperative Work and Social Computing** (CSCW 2018), New York, USA. 2018.

This paper investigates how often popular OSS projects hosted on GitHub adopt bots. We also report a qualitative analysis on how contributors and maintainers perceive the relevance of bot support. This served to motivate our research.

Paper 2. WESSEL, Mairieli; STEINMACHER, Igor; WIESE, Igor Scaliante; GEROSA, Marco Aurélio. *Should I Stale or Should I Close? An Analysis of a Bot that Closes Abandoned Issues and Pull Requests.* In: **1st International Workshop on Bots in Software Engineering** (BotSE 2019), Montréal, CA. 2019.

We report the results of a study that aimed at investigating the adoption of the *stale bot*, which helps maintainers triaging abandoned issues and pull requests. This also served to motivate our research.

Paper 3. WESSEL, Mairieli. *Leveraging Software Bots to Enhance Developers' Collaboration in Online Programming Communities.* In: **Companion Publication of the 2020 Conference on Computer Supported Cooperative Work and Social Computing: Doctoral Consortium**, 2020.

Paper 4. WESSEL, Mairieli. *Enhancing Developers' Support on Pull Requests Activities with Software Bots.* In: **ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) - Doctoral Symposium**, 2020.

Papers 3 and 4 are doctoral symposium papers that discuss the method for this dissertation and the expected contributions to the area of software bots in software engineering.

Papers related to Phase I – Investigating the Effects of Bot Adoption

Paper 5. WESSEL, Mairieli; SEREBRENIK, Alexander; STEINMACHER, Igor; WIESE, Igor Scaliante; GEROSA, Marco Aurélio. *What to Expect from Code Review Bots? A Survey with OSS Maintainers.* In: **Insightful Ideas and Emerging Results Track of Brazilian Symposium on Software Engineering (SBES - IIER)**, 2020.

This study reports how project maintainers experience code review bots. Our findings reveal that the most frequent expectations include enhancing the feedback bots provide to developers, reducing the maintenance burden for developers and enforcing code coverage.

Paper 6. WESSEL, Mairieli; SEREBRENIK, Alexander; STEINMACHER, Igor; WIESE, Igor Scaliante; GEROSA, Marco Aurélio. *Effects of Adopting Code Review Bots on Pull Requests to OSS Projects.* In: **36th IEEE International Conference on Software Maintenance and Evolution (ICSME)**, 2020.

In this study, we investigate how several activity indicators change after the adoption of a code review bot using regression discontinuity design. This paper received an *IEEE TCSE Distinguished Paper Award.*

Paper 7. WESSEL, Mairieli; SEREBRENIK, Alexander; STEINMACHER, Igor; WIESE, Igor Scaliante; GEROSA, Marco Aurélio. *Quality Gatekeepers: Investigating the Effects of Code Review Bots on Pull Request Activities.* **Submitted to Special Issue of ICSME 2020 of the Empirical Software Engineering (EMSE)**, 2021.

This paper extends paper 6. In this extended version, we provide a broader vision of how code review bots affect the pull request workflow, also considering the practical perspective of open-source developers.

Paper 8. KINSMAN, Timothy; WESSEL, Mairieli; GEROSA, Marco Aurélio; TREUDE, Christoph. *How Do Software Developers Use GitHub Actions to Automate Their Workflows?* In: **Mining Software Repositories Conference (MSR)**, 2021.

In this study, we investigate how several activity indicators change after the adoption of a GitHub Actions, following the studies conducted in Papers 6 and 7.

Papers related to Phase II – Cataloging Bot Interaction Challenges

Paper 9. WESSEL, Mairieli; STEINMACHER, Igor. *The Inconvenient Side of Software Bots on Pull Requests.* In: **2nd International Workshop on Bots in Software Engineering (BotSE)**, Seoul, South Korea. 2020.

This paper empirically investigates problems introduced by bots while interacting on pull requests. In this paper, we also introduced the concept of the meta-bot.

Paper 10. WESSEL, Mairieli; WIESE, Igor Scaliante; STEINMACHER, Igor; GEROSA, Marco Aurélio. *Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects.* In: **ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)**, 2021.

We identified several challenges caused by bots in pull request interactions. In particular, our findings indicate noise as a recurrent and central problem.

Papers related to Phase III – Designing Strategies to Overcome Noise

Paper 11. WESSEL, Mairieli; ABDELLATIF, Ahmad; SHIHAB, Emad; Igor; WIESE, Igor Scaliante; GEROSA, Marco Aurélio. *Bots for Pull Requests: The Good, the Bad, and the Promising.* **Under submission**.

In this paper, we applied Design Fiction as a participatory method to explore strategies to overcome the information overload caused by bots. Furthermore, it is worthwhile to mention that this paper was being prepared by the time this text was written.

1.4.2 Developed Software

BotHunter — we have been developing a tool to identify GitHub bots in collaboration with Dr. Emad Shihab and his team. This tool is useful for future research since we can identify the bots interacting on a project without searching it manually.

Curated list of SE bots — due to the lack of resource available of software bots on GitHub, we have created a curated list of software engineering bots.[3] The list was enhanced during the Dagstuhl Seminar "BOTse: Bots in Software Engineering".

1.4.3 Awards

Paper 7 received an *IEEE TCSE Distinguished Paper Award* at the International Conference on Software Maintenance and Evolution (ICSME 2020).

1.4.4 Funding

CNPq Scholarship. This research was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) grant 141222/2018-2.

ELAP 2021. The author's visit to Concordia University's Data-driven Analysis of Software (DAS) Lab, Canada, from February to July 2021 under the supervision of Dr.

[3] https://github.com/mairieli/awesome-se-bots

Emad Shihab. This exchange was supported by a scholarship from the Emerging Leaders in the Americas Program.

1.4.5 Advising

Graduation Thesis. Eric Patrick Delgado Ribeiro and Ronan Felipe Nascimento de Souza. *Agregada ou Separada? Como a Informação de Bots é Percebida por Desenvolvedores Novatos em Pull Requests.* 2021. Trabalho de Conclusão de Curso. Pontifícia Universidade Católica de Minas Gerais (PUC Minas). Advisors: Hugo Bastos de Paula / Mairieli Wessel.

1.4.6 Community Service

Because of the results of this research and its repercussion, the author was invited to:

Journal Guest Editor — Guest Editor of the Special Issue on Bots in Software Engineering, IEEE Software Magazine, 2021.

BotSE21 Co-organizer — Invited to the co-organize the 3^{rd} International Workshop on Bots in Software Engineering[4] co-located with ICSE 2021.

BotSE20 Program Committee — Invited to the program committee of the 2^{nd} International Workshop on Bots in Software Engineering co-located with ICSE 2020.

Dagstuhl Seminar — Invited to attend the Dagstuhl Seminar, “BOTse: Bots in Software Engineering”, [5] held at Schloss Dagstuhl in Germany from November 17 to November 22, 2019. Schloss Dagstuhl is one of the world’s leading research centers in informatics and has been hosting invitation-only seminars since 1990.

[4] https://botse.org

[5] https://www.dagstuhl.de/19471

Chapter 2

Background

In this chapter, we provide an overview of the origin and evolution of software bots. Additionally, we present a brief explanation of the definition that will be considered in this thesis. Then, we explore some existing approaches for classifying software bots, followed by the rise and adoption of bots on OSS projects. Finally, we also present an overview of the current state-of-the-art of software bots.

2.1 Origin and Evolution of Software Bots

The term *bot* originated as an abbreviation of *robot*. Since robots and software bots are used in the physical world and digital world respectively, the relationship between both terms emerged. The similarity between them is that robots and software bots are often used to automate repetitive tasks (C. Lebeuf, Zagalsky, *et al.*, 2019).

The origin of conversational agents (or *chatbots*), dates back to 1950 when Alan Turing proposed that machines could think (Turing, 1950). Turing's research triggered the development of chatbots, which are computer programs designed to interact with humans using a natural language (Shawar and Atwell, 2007). Initially, chatbots were created and used for entertainment. *Eliza* was the first computer program that enabled natural language interaction between humans and devices (Weizenbaum *et al.*, 1966). *Eliza* was developed in 1966 by an MIT professor called Joseph Weizenbaum, and was designed to mimic a psychotherapist. Since then, the interaction between computers and humans has been a challenge for researchers (Dale, 2016; Vinciarelli *et al.*, 2015; Zue and Glass, 2000).

Advancements in fields such as Artificial Intelligence (AI), Natural Language Processing (NPL), and Machine Learning (ML) have caused (i) a higher usage of bots in several domains; and (ii) partnerships in which computers and humans construct meaning around each other's activities (Farooq and Grudin, 2016). According to C. R. Lebeuf (2018), the mainstream adoption of software bots also occurred because of (i) numerous technological breakthroughs; (ii) dominant adoption of both messaging and voice-only platforms; (iii) and the abundance of public APIs and datasets. Bots enhance collaborative work (R Stuart Geiger, 2013) and influence changes in the workplace (Lee *et al.*, 2017).

Over the last few years, technological enterprises have developed bots as intelligent personal assistants, such as Apple's Siri (Winarsky *et al.*, 2012) and Google Assistant (Statt, 2016), using conversational interfaces to automate personal tasks for users. In contrast, thousands of bots perform specific tasks in a narrow domain of expertise (Dale, 2016). For example, bots have been used for education (Kerry *et al.*, 2008), focusing on students' engagement (Benotti *et al.*, 2014; Bii, 2013; Fryer *et al.*, 2017), self-guided learning (Pereira, 2016), course advising (Kim *et al.*, 2007), tutoring (Tamayo-Moreno and Pérez-Marín, 2017; Tegos and Demetriadis, 2017) and coaching (Maurer and Weihe, 2015).

Bots have also been used for marketing, e-commerce (Mimoun *et al.*, 2017; Thomas, 2016), and customer services (Gnewuch *et al.*, 2017; Jain *et al.*, 2018). Bots have also played a relevant role in peer production communities, such as Wikipedia (Cosley *et al.*, 2007; R Stuart Geiger, 2013; R. Stuart Geiger and Halfaker, 2017), Reddit (Long *et al.*, 2017), and social media (Savage *et al.*, 2016; Abokhodair *et al.*, 2015; Bin Xu *et al.*, 2014).

2.2 Software Bots Definition

Despite its increasing popularity, analyzing and understanding bots is a major challenge. The terminology used to describe software bots is vast, diverse, and often inconsistent (C. Lebeuf, Zagalsky, *et al.*, 2019). The terminology consists of different terms such as: robots, bots, chatbots, chatterbots, and software agents; which are often used for several fields, ranging from automated social media accounts to conversational agents. Consequently, this hinders a better understanding of the term *bot* and its usage.

Researchers and practitioners have defined software bots according to their specific applications. Following, we highlight four trends in how software bots have been described in the state-of-the-art.

Bots as automation providers. The definition of bots is often linked to their ability to automate tasks (Storey and Zagalsky, 2016; Cosley *et al.*, 2007; Long *et al.*, 2017). The Merriam-Webster dictionary defines bots as "*a computer program that performs automatic repetitive tasks.*" In the context of software development, Storey and Zagalsky (2016) proposed that bots are "*applications that perform repetitive predefined tasks to save developer's time and increase their productivity.*" Several researchers have defined bots based on their autonomy to perform tasks. According to Wyrich and Bogner (2019), "*a bot is intelligent software that acts (to some extent) autonomously to achieve a defined goal and offers functionality for interaction.*" Moreover, autonomous bots can often lead to mistakenly defining them as computer scripts. As stated by C. R. Lebeuf (2018), some bots are complex programs written in a compiled language, and thus unsatisfy the requirements for computer scripts.

Bots with conversational skills. Bots are also defined based on their ability to communicate using human language (Matthies *et al.*, 2019; Abdellatif *et al.*, 2020). Furthermore, Dale (2016) describes the relationship between humans and bots as "*achiev[ing] some result by conversing with a machine in a dialogic fashion, using natural language.*" Additionally, many researchers and practitioners use the term *chatbot* to refer to software bots, and vice versa. Although several popular bots do in fact have some sort of language capability, engaging in conversations is not required for software bots (C. R. Lebeuf, 2018; Paikari, Choi, *et al.*, 2019). More specifically, *chatbots* stand out from software bots because of their ability to communicate with users through human language.

Bots with human-like behaviors. According to the Oxford English Dictionary, bots can act and be perceived as humans: "*an autonomous program on a network (especially the Internet) which can interact with systems or users, especially one designed to behave like a player in some video games*". Similarly, Maus (2017) defines bots as "*automated or largely automated programs that interface with online platforms in largely the same way that a typical human would be expected to: they hold normal accounts, make connections, and post content.*" In the context of software development, Erlenhov, Oliveira Neto, *et al.* (2019) proposed the definition of which *DevBots*: "*an artificial software developer which is autonomous, adaptive, and has technical as well as social competence.*" Monperrus, Urli, *et al.* (2019) goes as far as claiming that "*we are now at the beginning of an exciting era where software bots will make contributions that are of similar nature than those by humans.*"

Bots as an interface between users and services. As described by Storey and Zagalsky (2016), bots implement *"a conduit or an interface between users and services."* Recently, C. R. Lebeuf (2018) explored the similarities between the aforementioned interpretations of bots, and then proposed an updated definition: software bot is "*an interface that connects users to services.*" According to C. R. Lebeuf (2018), this interface usually provides *"additional value (in the form of interaction style, automation, anthropomorphism, etc.) on top of the software service's basic capabilities.*" Moreover, the way bots act now, is just a different way to access services, which makes the difference between them so subtle.

In this thesis, we focus on *GitHub bots*. Although we are not intended to define software bots in general, it is important to narrow the definition of *GitHub bots* we adopted. Our definition builds upon many of the aforementioned definitions and reflects how a bot works on GitHub. Similarly to human users, *GitHub bots* have their own user profile and can behave as a developer: opening, closing or commenting on pull requests and issues. Playing a role within the development team, *GitHub bots* execute well-defined tasks that complement other developers' work. They also serve, for example, as an interface between developers and other tools, such as Continuous Integration (CI) tools.

Essentially, we define *GitHub bots* as **task-oriented bots** that act on the GitHub environment. A *GitHub bot* is an application that integrates its work with human tasks (Farooq and Grudin, 2016), serving as a conduit between users and services (Storey and Zagalsky, 2016). They provide new forms of interactions with already existing tools (Bradley *et al.*, 2018), automating predefined tasks and binding services together.

The following sections explore the classification of software bots in the context of software engineering, and reasons for their popularity in Github.

2.3 Bots on Software Engineering

In terms of understanding the practical implications of bot adoption, Storey and Zagalsky (2016) describe a cognitive framework to explain how bots support software development productivity. The framework identifies bots' role in different phases of the software development lifecycle: code bots, test bots, operations bots, support bots, and documentation bots. Storey and Zagalsky (2016) highlight the importance of automating tedious tasks during development by integrating bots in the developer's existing environment.

In addition to defining software bots, C. Lebeuf, Zagalsky, *et al.* (2019) provide a faceted taxonomy to characterize bots based on a literature review. The taxonomy consists of three main facets: (i) properties of the environment in which the bot was created; (ii) intrinsic properties of the bot; and (iii) the interactions of the bot within the environment. C. Lebeuf, Zagalsky, *et al.* (2019) detail the facets into sets of sub-facets.

Erlenhov, Oliveira Neto, *et al.*, 2019 propose a faceted taxonomy of bots focused on software development activities (*DevBots*), orthogonal to the taxonomy provided by C. Lebeuf, Zagalsky, *et al.* (2019). The taxonomy consists of four main facets: (i) purpose of the bot; (ii) type of initiation of the bot; (iii) properties of bot communication; and (iv) how bots handle information for completing tasks.

In terms of collaborative software engineering, preliminary studies aim to understand bots and their interactions with messaging tools (Lin *et al.*, 2016) and social media (Pérez-Soler *et al.*, 2017). In collaborative software development environments, bots automate tasks that generally require human interaction (C. Lebeuf, Storey, *et al.*, 2018). The following section focuses on the rise and adoption of software bots for supporting OSS development on GitHub.

2.3.1 The rise of bots on GitHub

Open Source Software (OSS) development is inherently collaborative, frequently involving a community of geographically dispersed developers (I. F. Steinmacher, 2015). These developers commonly work on social coding platforms, such as GitHub, that provide features for collaborating and sharing (Dabbish *et al.*, 2012). To receive external contributions, repositories are shared by *fork* (i.e., clone), and modified by pull requests. Figure 2.1 shows an overview of the process to send contributions using pull requests.

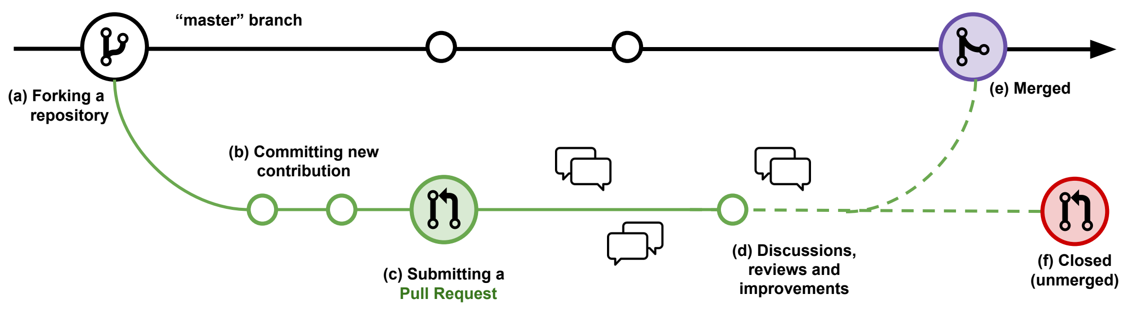

"master" branch
(a) Forking a repository
(b) Committing new contribution
(c) Submitting a Pull Request
(d) Discussions, reviews and improvements
(e) Merged
(f) Closed (unmerged)

Figure 2.1: *Pull-based model workflow on GitHub (Derived from GitHub)*

In the pull-based model, contributors lack access to modify repositories of their interest (Gousios, Pinzger, *et al.*, 2014a). Instead, contributors can clone (or *fork*) the main repository, as shown in Figure 2.1(a). Once the main repository has been forked, they can provide their contributions independently (Figure 2.1(b)), thus not interfering in other repositories. It is worth mentioning that contributions are not restricted to writing code, other types include reporting bugs, translating, improving documentation, and so on. Once changes are ready for submission, contributors *request a pull* of such changes to the main repository, as shown in Figure 2.1(c). The maintainers are then responsible for evaluating the quality of the contributions, offering suggestions, and discussing with the contributors (Figure 2.1(d)). If the contribution fulfills the project requirements, a maintainer merges it to the specified branch of the main repository (Figure 2.1(e)). Otherwise, the maintainer rejects and closes the pull request (Figure 2.1(f)).

The pull-based model offers new opportunities for community engagement, especially to OSS community, but at the same time increases the workload for maintainers to communicate, review code, deal with license issues, explain project guidelines, run tests, and merge pull requests (Gousios, Storey, *et al.*, 2016). Due to this intensive integration workload inherent to the pull request model (Gousios, Storey, *et al.*, 2016), software bots have become particularly relevant for OSS projects hosted on GitHub.

On GitHub, bots are commonly adopted to automate a variety of predefined tasks around pull requests. As aforementioned, GitHub bots have their own user profile and can open, close, or comment on pull requests and issues. Additionally, some bots have an official integration with GitHub and are available at the GitHub marketplace[1] or Probot [2]. These bots are properly tagged in the pull request messages, such as *Dependabot*[3].

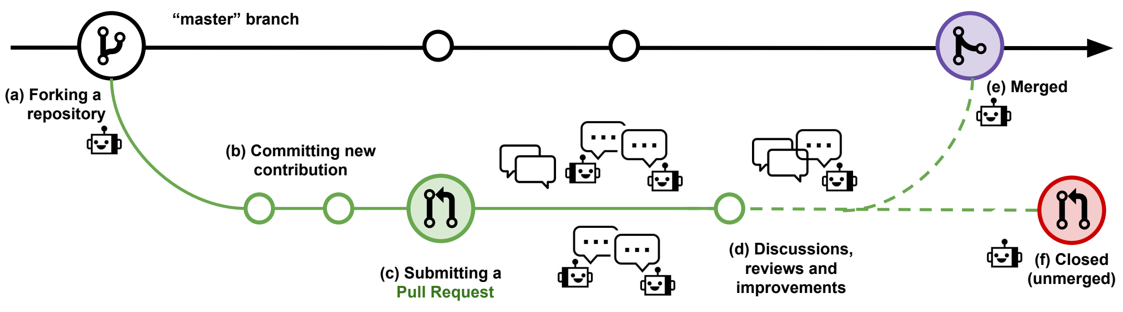

"master" branch
(a) Forking a repository
(b) Committing new contribution
(c) Submitting a Pull Request
(d) Discussions, reviews and improvements
(e) Merged
(f) Closed (unmerged)

Figure 2.2: *Pull-based model workflow integrated with bots*

Bots have spread across all pull-based workflow, as shown in Figure 2.2. Bots can be found submitting pull requests, interacting through comments, and merging or closing them. Some bots automatically check the project's source code, searching for broken dependencies, vulnerabilities, or bugs; and then submit a pull request for fixing these issues as shown in Figure 2.2(a). For example, *Dependabot* automatically creates pull requests to keep project dependencies up-to-date. The *Refactoring-Bot*, proposed by Wyrich and Bogner (2019), creates pull requests to refactor the code, removing code smells.

There are also bots designed to support contributors and maintainers after pull requests have been submitted, which aims to facilitate discussions and reviews (Figure 2.2(d)). *Git*

[1] https://github.com/marketplace

[2] https://probot.github.io/apps/

[3] https://dependabot.com

Enforcer[4], comments on pull requests that do not satisfy specific rules defined by the project maintainers. Moreover, it is also possible to communicate with some bots through comments in the pull requests. *Reminders*[5] bot, for example, sets a reminder on a specific pull request when the maintainer requests it. Other bots help maintainers closing or merging pull requests, as shown in both Figure 2.2(e) and Figure 2.2(f). *Stale bot*, studied in our previous work (Wessel, I. Steinmacher, *et al.*, 2019), automatically closes abandoned pull requests.

In fact, these new "*team members*" have become new voices on the pull request conversation (Monperrus, 2019). According to Brown and Parnin (2019), the human-bot interaction on GitHub can be inconvenient and lead to negative feedback due to poor design. Consequently, although bots can be useful for supporting maintainers' work, their comments are seen as spam, and sometimes are quickly ignored by contributors. Interviewing industry practitioners, Erlenhov, Neto, *et al.* (2016) found that bots cause interruption and noise, trust, and usability issues.

A similar problem is explored on Wikipedia community (Müller-Birn *et al.*, 2013; Zheng *et al.*, 2018). Wikipedia bots change the overall ecosystem by interacting with their operators, managers, and human editors, as well as other bots. For example, Zheng *et al.* (2018) 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 (Müller-Birn *et al.*, 2013). 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.

2.4 Related Work

This section presents an overview of the current state of software bots supporting software development activities, with an emphasis on bots integrated into the GitHub environment: *GitHub bots*.

2.4.1 Bots Supporting Software Development Activities

In recent years, software bots have been proposed to support collaborative software engineering, encompassing both technical and social aspects of software development activities (Lin *et al.*, 2016). Storey and Zagalsky (2016) present a cognitive framework to support the software bot landscape. Matthies *et al.* (2019) proposed employing bots in agile retrospectives, thereby attempting to enhance the development process. Matthies *et al.* (2019) consider a Slack bot as a convenient interface for interacting with the software development team.

Other studies have used bots for answering developers' questions, thus saving time and improving efficiency. For example, Bowen Xu *et al.* (2017) proposed *AnswerBot*, a

[4] https://github.com/Schachte/Git-Enforcer

[5] https://probot.github.io/apps/reminders/

bot that automatically generates an answer to a technical problem. Basically, *AnswerBot* extracts answers from Stack Overflow and summarizes them according to developers' questions. A user study was conducted with 6 developers to evaluate answers generated by *AnswerBot*. The results show that *AnswerBot* can potentially help developers to improve their efficiency for problem-solving.

Murgia *et al.* (2016) built and compared the impact of two identical bots, which aimed to answer developers' questions in Stack Overflow. The only difference between these two bots is their identity: the first bot is presented as a human being, while the second is presented as a bot. The results show that developers had a higher confidence in results provided by the "human" bot than its counterpart. Murgia *et al.* (2016) explain that developers have a very low tolerance and very high expectations for answers or artifacts provided by bots.

In order to automate data extraction from software repositories, Abdellatif *et al.* (2020) designed *MSRBot*, a bot that answers questions of a specific project. Abdellatif *et al.* (2020) performed a user study with 12 developers, who were invited to ask questions about specific topics. The results indicate that *MSRBot* is useful, efficient, and accurate for answering the most common questions. However, developers suggest deep-dive answers, thereby diving deeper into the results.

Similarly, Romero *et al.* (2020) also designed and implemented GitterAns, a bot to automatically detect when a developer asks a technical question in a Gitter chat and leverages the information from Q&A forums to provide the developer with possible answers to their question. According to Romero *et al.* (2020), a preliminary evaluation showed that GitterAns is currently able to detect troubleshooting questions with 78% accuracy.

Cerezo *et al.* (2019) proposed a bot to help developers find the most appropriate members of open source projects. In order to meet research objectives, a preliminary study was performed, analyzing interactions between participants. Moreover, interviewers and emotion tests were conducted. The results show that while participants are open to recommendations from bots, significant work is necessary to increase its acceptance. In fact, participants expected the bot to communicate with them, rather than simply answering queries.

Paikari, Choi, *et al.* (2019) implemented a prototype version of *Sayme*, a bot that provides information on potential direct and indirect conflicts, thereby helping developers to solve them. Unlike previous studies, *Sayme* was designed to operate both proactively and reactively. Proactively, this bot can respond to user questions about different works by developers. Reactively, *Sayme* can notify developers of emerging conflicts.

Dominic *et al.* (2020) proposed a conversational bot that would recommend projects to newcomers and assist in the onboarding to the open source community. The bot was designed to provide helpful resources, such as Stack Overflow related content. It would also be able to recommend human mentors for newcomers. Dominic *et al.* (2020) believe that having a bot to support newcomers may increase the chances of long-term engagement.

So far, the aforementioned studies focused on the design and development of bots, as well as the evaluation of bots' capabilities and performance. However, other studies in the literature investigate the usage and impact of software bots in communication platforms used by developers, such as Slack. Considering this perspective, LIN *et al.* (2016) conducted a survey to developers that used Slack or created Slack bots, thus aiming to understand their usage. Results demonstrated that developers use and create diverse Slack bots to support both technical and social activities. A. PINHEIRO *et al.* (2019) went a step further and conducted a survey to understand the motivations for developing a Slack bot. Results show that the main purpose of developers was to satisfy their personal needs, as well as to solve problems in either their workspace or daily life.

Although these studies are related to our research, they study bots that support development activities in online platforms in general, as known as *DevBots* (ERLENHOV, OLIVEIRA NETO, *et al.*, 2019). We summarize these studies in Table 2.1, which reveals that most of these bots were integrated into communication platforms, such as Slack and Discord. Furthermore, most of these bots focused on answering developers' questions about technical problems, repositories, or mentors. Section 2.4.2 presents studies specific to bots on GitHub.

Study	**Environment type**	**Study type**	**Main Findings**
MATTHIES *et al.* (2019)	Slack	Concept definition	Bots, more specifically *chatbots*, provide a convenient user interface for interacting with the outcomes of software activities.
Bowen XU *et al.* (2017) and CAI *et al.* (2019)	Standalone[1]	User study with 6 developers	*AnswerBot* can potentially help developers improve their efficiency for problem-solving.
ABDELLATIF *et al.* (2020)	Standalone[1]	User study with 12 developers	*MSRBot* significantly outperforms finding answers manually to developers questions.
ROMERO *et al.* (2020)	Gitter	Design and implementation	GitterAns achieved an accuracy of 0.78 in identifying technical questions
MURGIA *et al.* (2016)	Stack Overflow	User study	Developers had higher confidence in results provided by the "human" bot.
CEREZO *et al.* (2019)	Discord	User study with 6 participants	Participants expected the bot to communicate with them, rather than simply answering queries.
PAIKARI, CHOI, *et al.* (2019)	Slack	Design and implementation	*Sayme* was designed to operate proactively and reactively.
LIN *et al.* (2016)	Slack	Survey with (i) 53 developers that use Slack, and (ii) 51 developers that created Slack bots	Bots support both technical and social activities in Slack.
A. M. PINHEIRO *et al.* (2019) and A. PINHEIRO *et al.* (2019)	Slack	Survey with 43 bot developers	Developers create bots to satisfy their personal needs, and solve problems in either their workplace or daily life.
DOMINIC *et al.* (2020)	Standalone[1]	Concept definition	The bot aim at improving newcomers' experience by providing support not only during their first contribution, but by acting as an agent to engage them to the project.

[1] *According to C. R. LEBEUF (2018), standalone is when a bot is untied to a specific platform.*

Table 2.1: *Summary of studies on bots in Software Engineering*

2.4.2 Bots Supporting Pull Requests on GitHub

On GitHub, software bots are commonly adopted to automate a variety of predefined tasks, such as ensuring license agreement signing, reporting continuous integration failures, reviewing code and pull requests (Wessel, Souza, *et al.*, 2018; Peng, Yoo, *et al.*, 2018), triaging issues (Wessel, I. Steinmacher, *et al.*, 2019), repairing bugs (Urli *et al.*, 2018; Monperrus, 2019), refactoring source code (Wyrich and Bogner, 2019), recommending tools (Brown and Parnin, 2019), updating dependencies (Mirhosseini and Parnin, 2017), and fixing static analysis violations (Carvalho *et al.*, 2020). Moreover, several studies have been conducted to design and integrate bots into GitHub, helping developers gain productivity.

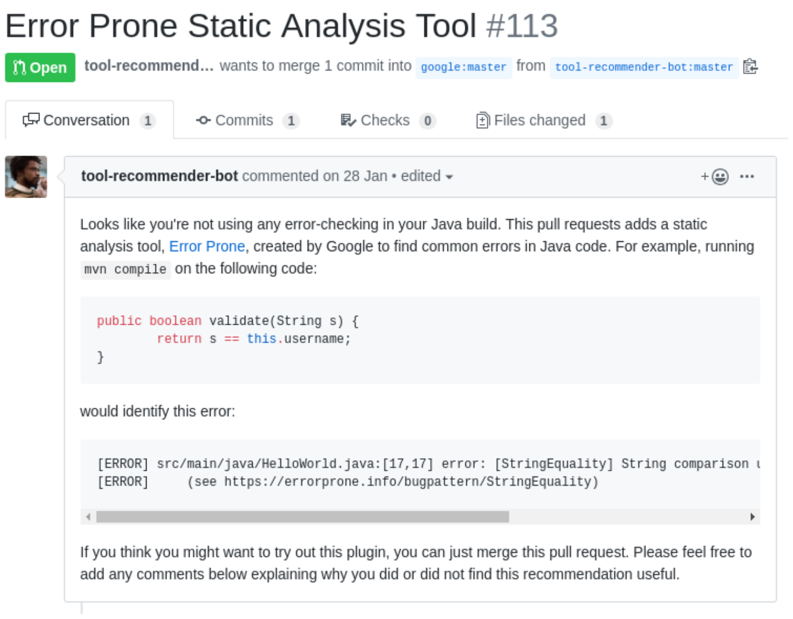
Urli *et al.* (2018) introduced *Repairnator*, a program repair bot that constantly monitors bugs during Continuous Integration (CI), and then tries to fix them by submitting a pull request. In the first round of experiments, the bot created pull requests for 15 different bugs using data from 11, 523 test failures of 1, 609 open-source software projects hosted on GitHub. Subsequently, the bot was enhanced to be human-competitive, meaning it aims to produce high-quality pull requests before humans. Monperrus, Urli, *et al.* (2018) reported that *Repairnator* was able to produce 5 pull requests that were successfully merged by the project's maintainers. Thus, *Repairnator* was able to fix bugs, that were considered of good quality, before humans themselves.

Although Urli *et al.* (2018) focus on repair bots, Tonder and Goues (2019) takes it to an even higher-level by extracting discussions from a particular language or repair techniques. Moreover, they discussed six principles for engineering repair bots based on syntax, semantics, and integration. Tonder and Goues (2019) express that the bot's effectiveness depends on its successful integration with developers' workflow.

Similar to previous studies, Wyrich and Bogner (2019) proposed the *Refactoring-Bot*, a bot that automatically refactors the code to remove code smells. This bot was designed to act autonomously, integrating into the natural workflow of the development team. After proposing code changes, the bot integrates them by submitting a pull request for maintainers to review. It is also possible to communicate with it through comments in the pull requests, allowing the maintainer to make small corrections to the code refactoring without having to switch to the development environment. Basically, *Refactoring-Bot* aims to eliminate the need for developers to manually find and correct code smells.

Bots have also been explored to improve tool adoption in software engineering. For example, Brown and Parnin (2019), proposed the *tool-recommender-bot*, a bot that provides tool recommendations to software developers. Initially, the bot only recommends static analysis tools. *tool-recommender-bot* automatically configures a project to use this tool and then submits a pull request with a generic message explaining how it works, as shown in Figure 2.3. Brown and Parnin (2019) applied *tool-recommender-bot* in real projects for evaluation purposes. Only two pull requests out of 52 recommendations were accepted. According to Brown and Parnin (2019), bots still need to find ways to interact with humans and handle all associated social and cognitive problems.

To reduce both development and maintenance efforts of redundant contributions, Ren *et al.* (2019) designed an approach to identify duplicated code changes in forks early. This


Error Prone Static Analysis Tool #113
Open
tool-recommend... wants to merge 1 commit into google:master from tool-recommender-bot:master
Conversation 1
Commits 1
Checks 0
Files changed 1
tool-recommender-bot commented on 28 Jan • edited

Looks like you're not using any error-checking in your Java build. This pull requests adds a static analysis tool, Error Prone, created by Google to find common errors in Java code. For example, running `mvn compile` on the following code:

```
public boolean validate(String s) {
        return s == this.username;
}
```

would identify this error:

```
[ERROR] src/main/java/HelloWorld.java:[17,17] error: [StringEquality] String comparison u
[ERROR]     (see https://errorprone.info/bugpattern/StringEquality)
```

If you think you might want to try out this plugin, you can just merge this pull request. Please feel free to add any comments below explaining why you did or did not find this recommendation useful.

Figure 2.3: *Example of recommendation from tool-recommender-bot (Brown and Parnin, 2019)*

approach extracts similarities between code changes, and builds a machine learning model to predict redundancies. Ren *et al.* (2019) envision that this approach can be implemented as a bot to support developers in two scenarios: (i) helping project maintainers to reduce their workload by monitoring incoming pull requests, and (ii) helping contributors to detect redundant development.

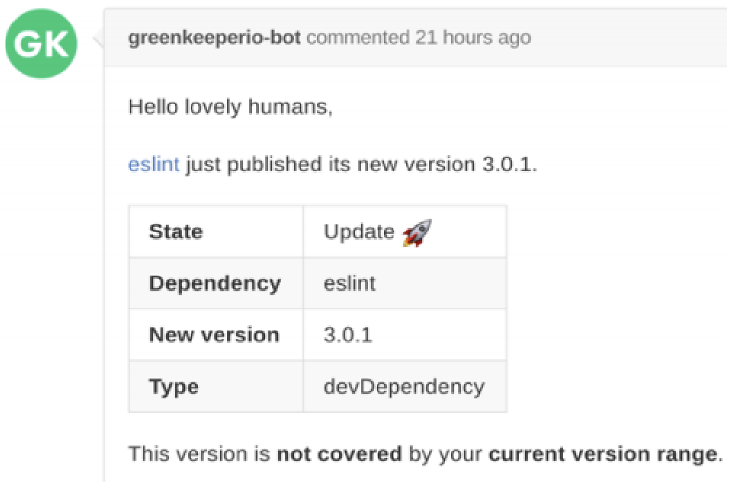
To integrate static analysis tools into developers' workflows, Carvalho *et al.* (2020) implemented a bot-based infrastructure. The proposed infrastructure, called the C-3PR bot, automatically proposes fixes to static analysis violations using pull requests. To evaluate the bot, Carvalho *et al.* (2020) monitored the bot activity in an industrial setting for a period of 8 months. They observed that, on average, the bot's pull requests are evaluated faster than pull requests created by humans. As a result of a focal group, the authors found that the C-3PR bot is efficient, reliable, and useful.

Unlike previous studies, Mirhosseini and Parnin (2017) analyzed 7,470 GitHub projects to understand whether badges, such as *David-DM*[6], or automated pull requests submitted by the *greenkeeper* bot[7] actually help maintainers to update outdated dependencies. Figure 2.4 shows an example of pull requests created by *greenkeeper* for updating specific dependencies. The results suggest that the *greenkeeper* bot can encourage project maintainers to update dependencies. On average, projects that used *greenkeeper* updated 1.6 times more than projects that did not use any tools. Although pull request notifications are useful, maintainers are often overwhelmed by notifications: only a third of pull requests were merged into the codebase.

Peng and Ma (2019) conducted a case study on how developers perceive and work with *mention bot*, which was developed by Facebook. Once a pull request is created, this

[6] https://david-dm.org

[7] https://greenkeeper.io


greenkeeperio-bot commented 21 hours ago

Hello lovely humans,

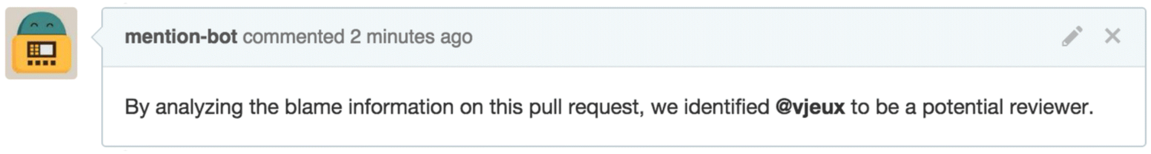
eslint just published its new version 3.0.1.

State	Update 🚀
Dependency	eslint
New version	3.0.1
Type	devDependency

This version is **not covered** by your **current version range**.

Figure 2.4: *Automated pull request created by greenkeeper.io. (Mirhosseini and Parnin, 2017)*

bot adds a recommendation as shown in Figure 2.5. Peng and Ma (2019) employed a mixed-method approach to investigate the usefulness of *mention bot*. First, they compared the pull requests before and after bot adoption. Then, they conducted a survey that involved contributors, reviewers, and project owners. The results show that even *mention bot* has saved developers' efforts; different user groups have different requirements for the bot. For example, project owners require simplicity and stability, contributors require transparency, and reviewers require selectivity. Additionally, results also show that developers are bothered with several review notifications during heavy workload.


mention-bot commented 2 minutes ago

By analyzing the blame information on this pull request, we identified **@vjeux** to be a potential reviewer.

Figure 2.5: *An example of the mention bot comments. (Peng and Ma, 2019)*

There are few approaches proposed to detect software bots on GitHub. For example, Dey *et al.* (2020) proposed BIMAN, an approach to detect software bots that commit code using commits meta-data (e.g., files modified by the commit). BIMAN achieved an AUC-ROC with 0.9 when it is evaluated in a dataset containing 461 bots. Golzadeh *et al.* (2021) proposed BoDeGHa, an approach to detect software bots in GitHub issues and pull requests comments based on their comments-related features (e.g., repetitive comment patterns). The authors evaluated the proposed approach on 5000 GitHub accounts. The results show that BoDeGHa achieved an F1-score of 0.98.

To help researchers integrate their new techniques into software development, Beschastnikh *et al.* (2017) envisioned the concept of an analysis bot platform called *Mediam*. The aim of *Mediam* is to help researchers upload their bots to the platform, and allow multiple developers to run it in GitHub, which will generate reports for feedback. Beschastnikh *et al.* (2017) envision bots being easily developed and deployed, allowing quick access to new methods developed by researchers. To avoid overwhelming developers, *Mediam* is responsible for deciding which notifications will be sent.

Table 2.2 summarizes the studies of bots on Github. Although these preliminary works focus on designing bots to be integrated into the GitHub workflow, little is known about the challenges they impose from the maintainers' and contributors' perspectives.

Study	**Bot type**	**Study type**	**Main Findings**
Urli *et al.* (2018), Monperrus, Urli, *et al.* (2018), and Monperrus, Urli, *et al.* (2019)	Repair bot	Design, Implementation and Empirical evaluation	*Repairnator* bot demonstrated program repair can be human-competitive: high-quality pull requests delivered before humans themselves found bugs.
Tonder and Goues (2019)	Repair bot	Design	Bot effectiveness depends on successful integration with human processes of software development.
Wyrich and Bogner (2019)	Code Refactoring bot	Design and Implementation	*Refactoring-bot* does not interrupt the developers' workflow, fixing code smells through pull requests.
Brown and Parnin (2019)	Recommendation bot	Design, Implementation and Empirical evaluation	Bots with simple technical knowledge alone are ineffective in influencing human behavior.
Ren *et al.* (2019)	Duplicate Development Detection bot	Design	Detecting duplicate development by using a bot could save contributors' and maintainers' efforts.
Mirhosseini and Parnin (2017)	Dependency Update bot	Case Study	Dependency update bots, such as greenkeeper, can encourage project maintainers to update dependencies.
Peng, Yoo, *et al.* (2018) and Peng and Ma (2019)	Mention Review bot	Case Study	Simplicity, stability, transparency and selectivity are critical to the user experiences regarding *mention bot.*
Beschastnikh *et al.*, 2017	Analysis bot	Design	Implementing the SE research solution as a bot have the potential to accelerate adoption by practitioners.
Carvalho *et al.*, 2020	Static Analysis bot	Design and Implementation	A bot-based infrastructure could mitigate some challenges that hinder the wide adoption of static analysis tools.
Dey *et al.* (2020)	Commit bots	Case Study	BIMAN achieved an AUC-ROC with 0.9
Golzadeh *et al.* (2021)	Issue and PR bots	Case Study	BoDeGHa achieved an F1-score of 0.98 detecting bot accounts

Table 2.2: *Summary of studies regarding bots on GitHub*

2.5 Final Considerations

In this chapter, we presented a historical perspective of software bots. Additionally, we proposed a definition of software bots on GitHub based on how researchers and practitioners have defined bots. In summary, we define *GitHub bot* as a specific category of software bot: a task-oriented bot, responsible for automating well-defined tasks on GitHub repositories. Similarly to human developers, these bots have their own user profile and interact through comments on pull requests.

The emergence of bot activity all over the OSS community on GitHub is an indication of the growing importance of these new team members for automating activities around pull requests. Despite this widespread adoption, bots are bothering both contributors and maintainers. Considering this, we aim to raise specific and actual interaction problems introduced by bots around pull requests, with an understanding of their context, and consequences. Ultimately, we aim to give support to mitigate some of these problems.

We also described the current state of bots supporting software development activities. Software bots on GitHub are different from those found supporting development activities in general. While GitHub bots automate specific tasks on pull requests, interacting with developers through comments, software engineering bots focus on answering developers' questions and are generally integrated into communication platforms.

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 (2019) report that bots still need to overcome problems such as notification workload. Mirhosseini and Parnin (2017) 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 (2019) 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 (2019), 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 effects and challenges that bots bring to interactions on social coding platforms. This dissertation 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.

In the following chapter, we describe our exploratory study dedicated to identifying the effects of adopting bots open-source projects' to pull requests.

Chapter 3

Effects of Adopting Bots on Pull Requests

To understand how the dynamics of GitHub project pull requests change following the adoption of bots, we investigate several activity indicators changes. We employed a regression discontinuity design on software projects from GitHub. More specifically, we used a mixed-methods approach (Easterbrook *et al.*, 2008) with a sequential explanatory strategy (Creswell, 2003), combining data analysis of GitHub data with semi-structured interviews conducted with open-source developers. We used a *Regression Discontinuity Design* (RDD) (Thistlethwaite and D. T. Campbell, 1960) to model the effects of code review bot adoption across 1,194 OSS projects hosted on GitHub. Afterwards, we conducted semi-structured interviews with 12 practitioners.

The results presented in this Chapter were partially published in different venues. The exploratory case study and the statistical analysis of code review bots effects were published at the 36th International Conference on Software Maintenance and Evolution (ICSME) (Wessel, Serebrenik, I. S. Wiese, *et al.*, 2020), and the extension that attempted to investigate the maintainers perceptions was submitted to the Special Issue of ICSME 2020 of the Empirical Software Engineering (EMSE) (Wessel, Serebrenik, I. Wiese, I. Steinmacher, and Marco A Gerosa, 2021). The reader may refer to these publications for additional details on each study.

3.1 Exploratory Case Study

As little is known about the effects of code review bots' adoption in the dynamics of pull requests, we conducted an exploratory case study (Runeson and Höst, 2009; Yin, 2003) to formulate hypotheses to further investigate in our main study.

To carry out our exploratory case study, we selected two projects that we were aware of that used code review bots for at least a one year: the Julia programming language project[1]

[1] https://github.com/JuliaLang/julia

and CakePHP,[2] a web development framework for PHP. Both projects have popular and active repositories—Julia has more than 26.1k stars, 3.8k forks, 17k pull requests, and 46.4k commits; while CakePHP has more than 8.1k stars, 3.4k forks, 8.6k pull requests, 40.9k commits, and is used by 10k projects. Both projects adopt a code review bot named Codecov, which posted the first comments on pull requests to the Julia project in July 2016 and CakePHP in April 2016.

After selecting the projects, we analyzed data from one year before and one year after bot adoption, using the data available in the GHTorrent dataset (Gousios and Spinellis, 2012). During this time frame, the only bot adopted by Julia and CakePHP was the Codecov bot. Similar to previous work (Zhao *et al.*, 2017), we exclude 30 days around the bot's adoption to avoid the influence of instability caused during this period. Afterward, we aggregated individual pull request data into monthly periods, considering 12 months before and after the bot's introduction. We choose the month time frame based on previous literature (Zhao *et al.*, 2017; David Kavaler *et al.*, 2019; Cassee *et al.*, 2020). All metrics were aggregated based on the month of the pull request being closed/merged.

We considered the eight activity indicators:

Merged/non-merged pull requests: the number of monthly contributions (pull requests) that have been merged, or closed but not merged into the project, computed over all closed pull requests in each time frame.

Comments on merged/non-merged pull requests: the median number of monthly comments—excluding bot comments—computed over all merged and non-merged pull requests in each time frame. We used the median because the distribution is skewed.

Time-to-merge/time-to-close pull requests: the median of monthly pull request latency (in hours), computed as the difference between the time when the pull request was closed and the time when it was opened. The median is computed using all merged and non-merged pull requests in each time frame. We used the median because the distribution is skewed.

Commits of merged/non-merged pull requests: the median of monthly commits computed over all merged and non-merged pull requests in each time frame. We use the median because the distribution is skewed.

We ran statistical tests to compare the activity indicators distributions before and after the bot adoption. As the sample is small, and there is no critical mass of data points around the bot's introduction, we used the non-parametric Mann-Whitney-Wilcoxon test (Wilks, 2011). In this context, the null hypothesis (H_0) is that the distributions pre- and post-adoption are the same, and the alternative hypothesis (H_1) is that these distributions differ. We also used Cliff's Delta (Romano *et al.*, 2006) to quantify the difference between these groups of observations beyond p-value interpretation. Moreover, we inspected the monthly distribution of each metric to search for indications of change.

As aforementioned, the case studies helped us to formulate hypotheses for the main study, which comprised more than one thousand projects. We formulated hypotheses

[2] https://github.com/cakephp/cakephp

whenever we observed changes in the indicators for at least one of the two projects we analyzed in the case study.

3.1.1 Case Study Results

In the following, we discuss the trends in project activities after bot adoption. We report the results considering the studied pull request activities: number of merged and non-merged pull requests, median of pull request comments, time-to-merge and time-to-close pull requests, and median of pull request commits.

Trends in the number of Merged and Non-merged Pull Requests

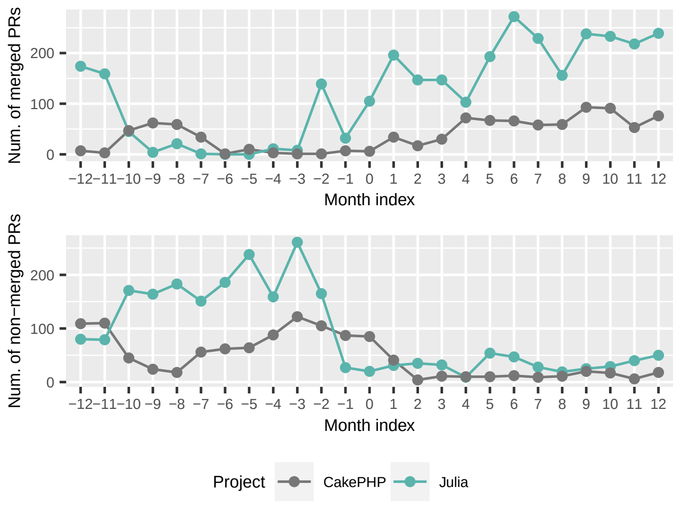


Figure 3.1: *Monthly merged and non-merged pull requests.*

The number of merged pull requests *increased* for both projects (Julia: p-value 0.0003, $\delta = -0.87$; CakePHP: p-value 0.001, $\delta = -0.76$), whereas the non-merged pull requests *decreased* for both projects (Julia: p-value 0.00007, $\delta = 0.87$; CakePHP: p-value 0.00008, $\delta = 0.95$). Figure 3.1 shows the monthly number of merged and non-merged pull requests, top and bottom respectively, before and after bot adoption for both projects. Based on these findings, we hypothesize that:

$H_{1.1}$ The number of monthly merged pull requests increases after the introduction of a code review bot.

$H_{1.2}$ The number of monthly non-merged pull requests decreases after the introduction of a code review bot.

Trends in the median of pull request comments

Figure 3.2 shows the monthly median of comments on merged and non-merged pull requests, respectively. CakePHP showed statistically significant differences between pre-

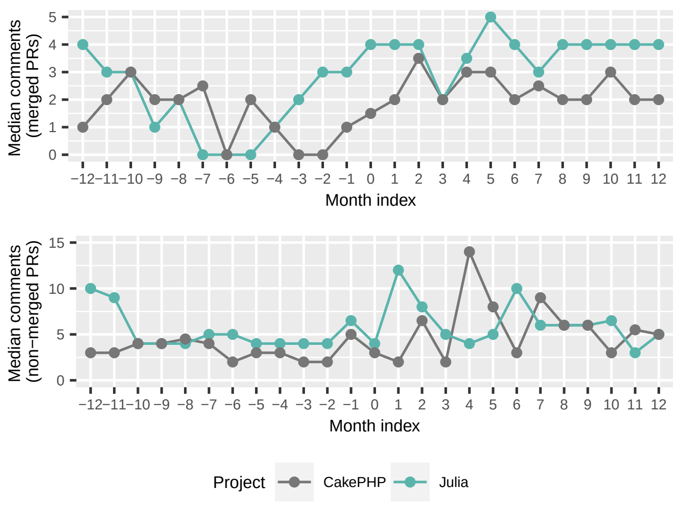


Figure 3.2: *Monthly comments on merged and non-merged pull requests.*

and post-adoption distributions. The number of comments *increased* for merged pull requests (p-value=0.01, δ = −0.56) and also for non-merged ones (p-value=0.03, δ = −0.50) with a large effect size. Thus, we hypothesize that:

***$H_{2.1}$* The adoption of code review bots is associated with an increase in the monthly number of comments for merged pull requests.**

***$H_{2.2}$* The number of monthly comments on non-merged pull requests increases after the adoption of a code review bot.**

Trends in the time to close pull request comments

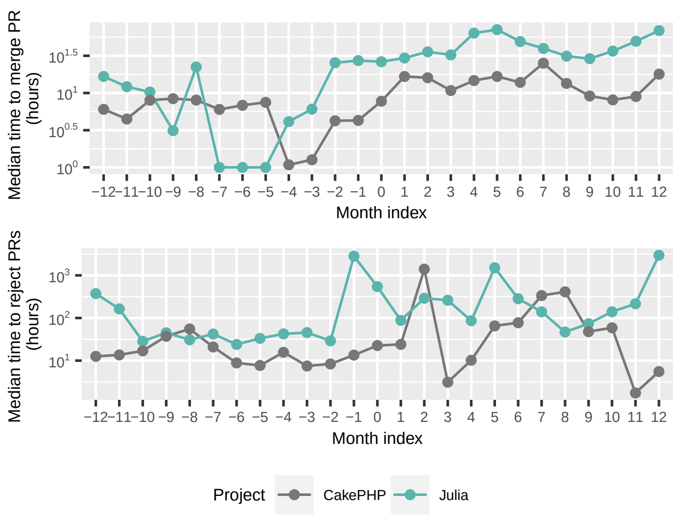


Figure 3.3: *Monthly median time to merge and reject pull requests.*

The median time to merge pull requests *increased* for both projects (Julia: p-value 0.0003, $\delta = -1.00$; CakePHP: p-value 0.000001, $\delta = -0.98$). Considering non-merged pull requests, the difference between pre- and post-adoption is statistically significant only for Julia. For this project, the median time to close pull requests *increased* (p-value 0.00007) with a large effect size ($\delta = -0.65$). The distribution can be seen in Figure 3.3. Therefore, we hypothesize that:

***$H_{3.1}$* There is an increase in the monthly time to merge pull requests after the introduction of code review bots.**

***$H_{3.2}$* There is an increase in the monthly time to reject pull requests after the adoption of a code review bot.**

Trends in the median of pull request commits

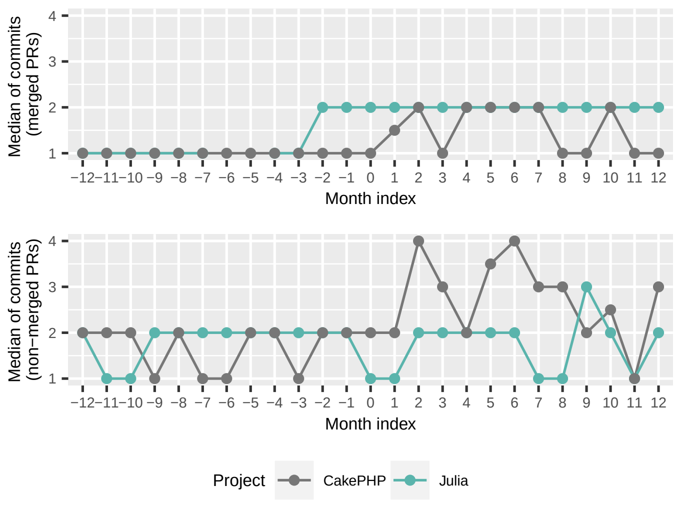


Figure 3.4: *Monthly commits on merged and non-merged pull requests.*

Investigating the number of pull request commits per month (see Figure 3.4), we note that the medians before the adoption are quite stable, especially for merged pull requests. In comparison, after adoption we observe more variance. The difference is statistically significant only for CakePHP, for which the number of pull request commits increased for merged pull requests (p-value=0.002, $\delta = -0.58$) and for non-merged pull requests (p-value=0.002, $\delta = -0.69$) with a large effect size. Based on this, we posit:

***$H_{4.1}$* There is an increase in the monthly number of commits for merged pull requests after code review bot adoption.**

***$H_{4.2}$* There is an increase in the monthly number of commits for non-merged pull requests after code review bot adoption.**

Summary of the Case Study. We observe statistically significant differences for all four activity indicators we investigated in at least one of the two projects. Based on these observations, we formulated hypotheses to be further investigated in our main study, comprising a large number of projects, and employed the regression discontinuity design.

3.2 Main Study Design

In this section, we describe our research questions (Section 3.2.1), the statistical approach and data collection procedures (Section 3.2.2), and the qualitative approach (Section 3.2.3).

3.2.1 Research Questions

The main goal of this study is to investigate how and for what reasons, if any, the adoption of code review bots affects the dynamics of GitHub project pull requests. To achieve this goal, we investigated the following research questions:

RQ1. *How do pull request activities change after a code review bot is adopted in a project?*

We investigate changes in project activity indicators, such as the number of pull requests merged and non-merged, number of comments, the time to close pull requests, and the number of commits per pull request. Using time series analysis, we account for the longitudinal effects of bot adoption. We also go one step further, exploring a large sample of open-source projects and focusing on understanding the effects of a specific bot category.

RQ2. *How could the change in pull request activities be explained?*

Besides understanding the change incurred by bot adoption, we explore why it happens. To do so, we interviewed a set of open-source developers who actually have been using these bots.

Figure 3.5 illustrates an overview of the steps taken to address the research questions. Next, we explain each step in order to justify the study design decisions.

3.2.2 Stage 1—Statistical Approach

Considering the hypotheses formulated in the case study, in our main study we employed time series analysis to account for the longitudinal effects of bot adoption. We employed Regression Discontinuity Design (RDD) (Thistlethwaite and D. T. Campbell, 1960; Imbens and Lemieux, 2008), which has been applied in the context of software engineering in the past (Zhao *et al.*, 2017; Cassee *et al.*, 2020). RDD is a technique used to model the extent of a discontinuity at the moment of intervention and long after the intervention. The technique is based on the assumption that if the intervention does not

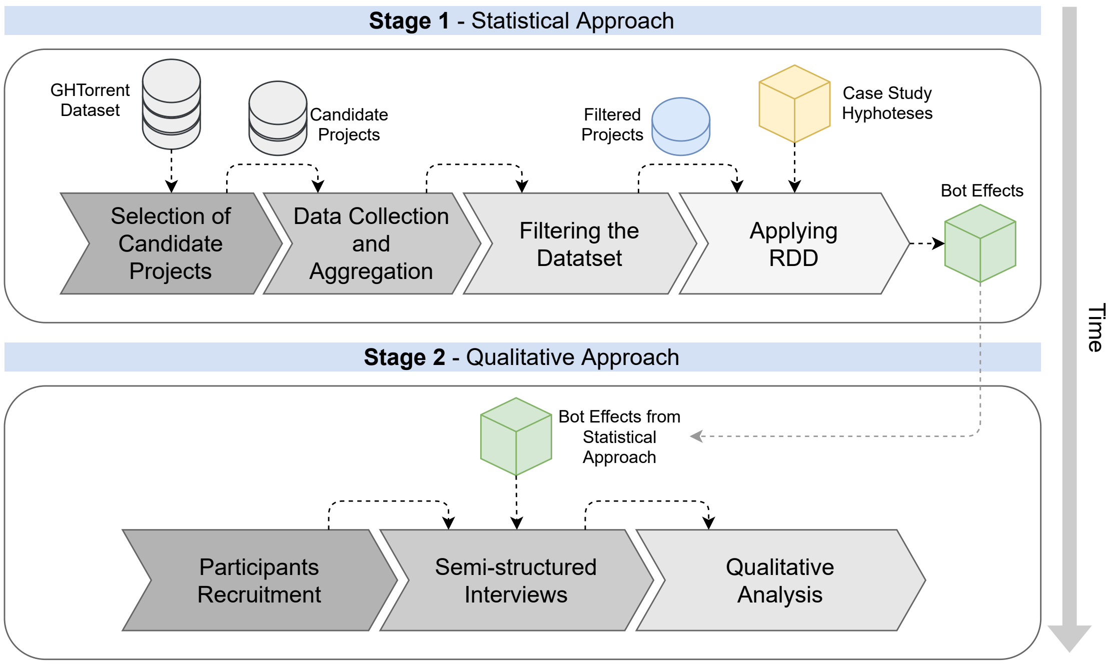

Stage 1 - Statistical Approach
GHTorrent Dataset
Candidate Projects
Filtered Projects
Case Study Hyphoteses
Bot Effects
Selection of Candidate Projects
Data Collection and Aggregation
Filtering the Datatset
Applying RDD
Stage 2 - Qualitative Approach
Bot Effects from Statistical Approach
Participants Recruitment
Semi-structured Interviews
Qualitative Analysis
Time

Figure 3.5: *Main Research Design Overview.*

affect the outcome, there would be no discontinuity, and the outcome would be continuous over time (Cook and D. Campbell, 1979). The statistical model behind RDD is

$$\begin{aligned} y_i = \alpha + \beta \cdot time_i + \gamma \cdot intervention_i + \\ \delta \cdot time\_after\_intervention_i + \eta \cdot controls_i + \varepsilon_i \end{aligned}$$

where i indicates the observations for a given project. To model the passage of time as well as the bot introduction, we include three additional variables: *time*, *time after intervention*, and *intervention*. The *time* variable is measured as months at the time j from the start to the end of our observation period for each project (24 months). The *intervention* variable is a binary value used to indicate whether the time j occurs before ($intervention = 0$) or after ($intervention = 1$) adoption event. The *time_after_intervention* variable counts the number of months at time j since the bot adoption, and the variable is set up to 0 before adoption.

The $controls_i$ variables enable the analysis of bot adoption effects, rather than confounding the effects that influence the dependent variables. For observations before the intervention, holding controls constant, the resulting regression line has a slope of β, and after the intervention $\beta + \delta$. The size of the intervention effect is measured as the difference equal to γ between the two regression values of y_i at the moment of the intervention.

Considering that we are interested in the effects of code review bots on the monthly trend of the number of pull requests, number of comments, time-to-close pull requests, and number of commits over a pull request, and all these for both merged and non-merged pull requests, we fitted eight models (2 cases x 4 variables). To balance false-positives and false-negatives, we report the corrected p-values after applying multiple corrections using the method of Benjamini and Hochberg (Benjamini and Hochberg, 1995). We implemented the RDD models as a mixed-effects linear regression using the R package *lmerTest* (Kuznetsova *et al.*, 2017).

To capture project-to-project and language-to-language variability, we modeled *project name* and *programming language* as random effects (Gałecki and Burzykowski, 2013). By modeling these features as random effects, we can account for and explain different behaviors observed across projects or programming languages (Zhao *et al.*, 2017). We evaluate the model fit using *marginal* (R^2_m) and *conditional* (R^2_c) scores, as described by Nakagawa and Schielzeth (Nakagawa and Schielzeth, 2013). The R^2_m can be interpreted as the variance explained by the fixed effects alone, and R^2_c as the variance explained by the fixed and random effects together.

In mixed-effects regression, the variables used to model the intervention along with the other fixed effects are aggregated across all projects, resulting in coefficients useful for interpretation. The interpretation of these regression coefficients supports the discussion of the intervention and its effects, if any. Thus, we report the significant coefficients ($p < 0.05$) in the regression as well as their variance, obtained using ANOVA. In addition, we *log* transform the fixed effects and dependent variables that have high variance (Sheather, 2009). We also account for multicollinearity, excluding any fixed effects for which the variance inflation factor (VIF) is higher than 5 (Sheather, 2009).

Selection of Candidate Projects To identify open-source software projects hosted on GitHub that at some point had adopted a code review bot, we queried the GHTorrent dataset (Gousios and Spinellis, 2012) and filtered projects in which at least one pull request comment was made by one of the code review bots identified by Wessel et al. (Wessel, Souza, *et al.*, 2018). Following the method used by Zhao et al. (Zhao *et al.*, 2017) to assemble a time series, we considered only those projects that had been active for at least one year before and one year after the bot adoption. We found 4,767 projects that adopted at least one of the code review bots. For each project, we collected data on all its merged and non-merged pull requests.

Data Collection and Aggregation Similar to the exploratory case study (see Section 3.1), we aggregated the project data in monthly time frames and collected the four variables we expected to be influenced by the introduction of the bot: number of merged and non-merged pull requests, median number of comments, median time-to-close pull requests, and median number of commits. All these variables were computed over pull requests that have been merged and non-merged in a time frame.

We also collected six control variables, using the GHTorrent dataset (Gousios and Spinellis, 2012):

Project name: the name of the project, used to identify the project on GitHub. We accounted for the fact that different projects can lead to different contribution patterns. We used the project name as a random effect.

Programming language: the primary project programming language as automatically determined and provided by GitHub. We considered that projects with different programming languages can lead to different activities and contribution patterns (Zhao *et al.*, 2017;

Cassee *et al.*, 2020). We used programming language as a random effect.

Time since the first pull request: in months, computed since the earliest recorded pull request in the entire project history. We use this to capture the difference in adopting the bot earlier or later in the project life cycle, after the projects started to use pull requests (Zhao *et al.*, 2017; Cassee *et al.*, 2020).

Total number of pull request authors: as a proxy for the size of the project community, we counted how many contributors submitted pull requests to the project.

Total number of commits: as a proxy for the activity level of a project, we computed the total number of commits.

Number of pull requests opened: the number of contributions (pull requests) received per month by the project. We expected that projects with a high number of contributions also observe a high number of comments, latency, commits, and merged and non-merged contributions.

Bot name	**GitHub user**	**# of projects**
Ansible's issue bot	ansibot	1
Elastic Machine	elasticmachine	3
Codecov	codecov-io	460
Coveralls	coveralls	730
		Total of 1, 194 under study

Table 3.1: *An overview of the studied bots*

Filtering the final dataset After excluding the period of instability (30 days around the adoption), we inspected the dataset and found 223 projects with no comments authored by any of the studied bots. We manually checked 30% of these cases and concluded that some projects only added the bot for a testing period and then disabled it. We removed these 223 projects from our dataset.

We also checked the activity level of the candidate projects, since many projects on GitHub are inactive (Gousios, Pinzger, *et al.*, 2014b). We excluded from our dataset projects without at least a six month period of consistent pull request activity during the one-year period before and after bot adoption. After applying this filter, a set of 1, 740 GitHub software projects remained. To ensure that we observed the effects of each bot separately, we also excluded from our dataset 78 projects that adopted more than one of the studied bots and 196 projects that used non-code review bots. In addition, we checked the activity level of the bots on the candidate projects. We excluded 272 projects that had not received any comments during the previous four months. After applying all filters, 1, 194 GitHub software projects remained. Table 3.1 shows the number of projects per bot. All of these four bots perform similar tasks on pull requests—providing comments on pull requests about code coverage.

3.2.3 Stage 2—Qualitative approach

As aforementioned, we also applied a qualitative approach aimed to understand the effects evidenced by the statistical approach from the practitioners' perspective. In the following, we describe the participants recruitment, semi-structured interview procedures, and the qualitative analysis.

Participants recruitment In this study, we employed several strategies to recruit participants. First, we advertised the interview on social media platforms frequently used by developers (Singer *et al.*, 2014; Storey, Treude, *et al.*, 2010; Aniche *et al.*, 2018), including Twitter, Facebook, and Reddit. We also manually searched the projects that were part of the statistical analysis for pull requests explicitly installing or (re)configuring the analyzed bots. We added a comment on some of these pull requests to invite the pull request author to the interview. We also sent emails to personal contacts who we knew had experience with these bots. In addition, we asked participants to refer us to other qualified participants.

We continued recruiting participants till we came to an agreement that the last three interviews had not provided any new findings. According to Strauss and Corbin (A. Strauss and Juliet M Corbin, 1997), sampling can be discontinued once the data collection no longer unveils new information. Additionally, the size of our participant set is in line with the anthropology literature, which mentions that a set of 10-20 knowledgeable people is sufficient to uncover and understand the core categories in any study of lived experience (Bernard, 2017).

Participants Demographics In total, we interviewed 12 open-source developers experienced with code review bots—identified here as P1–P12. Out of these twelve participants, one is an open-source maintainer, two are contributors, and the other nine are both maintainers and contributors. In addition, participants are geographically distributed across Europe (≃50%), North America (≃25%), and South America (≃25%). Snowballing was the origin of five of our participants. Personal contacts was the origin of four of our participants. The advertisements on social media were the origin of the other three interviews. Table 4.1 presents the demographic information of the interviewees.

Participant ID	OSS Experience (years)	Experienced with bots as		Location	Gender
		Maintainer	Contributor		
P1	4-5	✓		North America	Man
P2	Over 10	✓	✓	North America	Man
P3	4-5	✓	✓	Europe	Man
P4	3	✓	✓	Europe	Man
P5	4-5	✓	✓	Europe	Woman
P6	Over 10	✓	✓	North America	Man
P7	5-10	✓	✓	Europe	Man
P8	4-5	✓	✓	Europe	Man
P9	1		✓	Europe	Man
P10	4-5	✓	✓	South America	Man
P11	4-5	✓		South America	Man
P12	Over 10		✓	South America	Man

Table 3.2: *Demographics of interviewees*

Semi-structured interviews

We conducted *semi-structured* interviews, comprising open- and closed-ended questions designed to elicit foreseen and unexpected information and enable interviewers to explore interesting topics that emerged during the interview (Hove and Anda, 2005). Before each interview, we shared a consent form with the participants asking for their agreement. By participants' requests, one interview (P11) was conducted via email. The other eleven interviews were conducted via video calls. The participants received a 25-dollar gift card as a token of appreciation for their time.

We started the interviews with a short explanation of the research objectives and guidelines, followed by demographic questions to capture the familiarity of the interviewees with open-source development and code review bots. We then described to the interviewee the study we conducted and the main findings from the statistical approach and asked the developers to conjecture about the reasons for the effects we observed:

Q1. After adopting a code review bot there are more merged pull requests, less communication between developers, fewer rejected pull requests, and faster rejections. We are intrigued about these effects and would like to hear thoughts from developers who actually use these bots. Could you conjecture the reasons why this happens?

We follow-up this question with more specific questions when participants have not mentioned reasons for any of the four observed effects. Afterwards, we asked two additional questions:

Q2. Have you observed these effects in your own project?

Q3. What other effects did you observe in your project and attribute to the introduction of the code review bot?

The detailed interview script is publicly available[3]. Each interview was conducted remotely and lasted, on average, 35 minutes.

Qualitative analysis of interviews

Each interview recording was transcribed by the first author of this paper. We then analyzed the interview transcripts by applying open and axial coding procedures (A. L. Strauss and J. M. Corbin, 1998; Stol *et al.*, 2016) throughout multiple rounds of analysis. We started by applying open coding, whereby we identified the reasons for bots' effects. To do so, the author of this dissertation conducted a preliminary analysis, identifying the main codes. Then, the author discussed with two other experienced researchers the coding in weekly hands-on meetings. These discussions aimed to increase the reliability of the results and mitigate bias (Patton, 2014). Afterwards, the author further analyzed and revised the interviews to identify relationships between concepts that emerged from the open coding analysis (axial coding). During this process, we employed a constant comparison method (Glaser and Anselm L Strauss, 2017), wherein we continuously compared the results from one interview with those obtained from the previous ones.

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

3.3 Main Study Results

In the following, we report the results of our study by research question.

3.3.1 Effects of Code Review Bot Adoption (RQ1)

In this section, we discuss the effects of code review bot adoption on project activities along four dimensions: (i) accepted and rejected pull requests, (ii) communication, (iii) pull request resolution efficiency, and (iv) modification effort.

Effects in Merged and Non-merged Pull Requests

We start by investigating the effects of bot adoption on the number of merged and non-merged pull requests. From the exploratory case study, we hypothesized that the use of code review bots is associated with an increase in the number of monthly merged pull requests and a decrease in the number of monthly non-merged pull requests. We fit two mixed-effect RDD models, as described in Section 3.2.2. For these models, the *number of merged/non-merged pull requests* per month is the dependent variable. Table 3.3 summarizes the results of these two RDD models. In addition to the model coefficients, the table also shows the SS, with a variance explained for each variable.

	Merged Pull Requests		Non-merged Pull Requests	
	Coefficients	SS	Coefficients	SS
Intercept	-0.262***		-0.574***	
TimeSinceFirstPullRequest	0.00004**	4.3	-0.0001***	2.4
log(TotalPullRequestAuthors)	-0.094***	171.8	0.086***	775.7
log(TotalCommits)	0.042***	484.0	0.068***	428.6
log(OpenedPullRequests)	0.494***	8227.1	0.388***	4958.5
log(PullRequestComments)	0.433***	2954.3	0.389***	2341.0
log(PullRequestCommits)	0.272***	721.0	0.165***	255.5
time	0.004***	203.2	-0.004***	376.1
interventionTrue	0.095***	16.8	-0.163***	48.4
time_after_intervention	0.004**	1.7	-0.004**	1.6
Marginal R^2		0.68		0.67
Conditional R^2		0.75		0.74

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$. *SS stands for "Sum of Squares".*

Table 3.3: *The Effects of Code Review bots on PRs. The response is* ***log(number of merged/non-merged PRs)*** *per month.*

[3] https://doi.org/10.5281/zenodo.4618498

Analyzing the model for merged pull requests, we found that the fixed-effects part fits the data well ($R^2_m = 0.68$). However, considering $R^2_c = 0.75$, variability also appears from project-to-project and language-to-language. Among the fixed effects, we observe that the number of monthly pull requests explains most of the variability in the model. As expected, this indicates that projects receiving more contributions tend to have more merged pull requests, with other variables held constant.

Furthermore, the statistical significance of the time series predictors indicates that the adoption of code review bots affected the trend in the number of merged pull requests. We note an increasing trend before adoption; a statistically significant discontinuity at adoption; and a positive trend after adoption that indicates that the number of merged pull requests increased even faster.

Similar to the previous model, the fixed-effect part of the non-merged pull requests model fits the data well ($R^2_m = 0.67$), even though a considerable amount of variability is explained by random effects ($R^2_c = 0.74$). We note similar results on fixed effects: projects receiving more contributions tend to have more non-merged pull requests. All time-series predictors for this model are statistically significant, showing a measurable effect of the code review bot's adoption on the time to review and accept a pull request. We note a decreasing trend before adoption, a statistically significant discontinuity at the adoption time, and a slight acceleration after adoption in the decreasing time trend seen before adoption.

Therefore, based on models for merged and non-merged pull requests, we confirm both $\mathbf{H}_{1.1}$ and $\mathbf{H}_{1.2}$.

Effects in Merged and Non-merged Pull Requests. Overall, there are more monthly merged pull requests and fewer monthly non-merged pull requests after adopting a code review bot.

Effects on Developers' Communication

In the exploratory case study, we hypothesized that bot adoption increases monthly human communication on pull requests for both merged and non-merged pull requests. To statistically investigate this, we fit one model to merged pull requests and another to non-merged ones. The *median of pull request comments* per month is the dependent variable, while *number of monthly pull requests*, *median of time-to-close pull requests*, and *median of pull request commits* are independent variables. Table 3.4 shows the results of the fitted models.

Considering the model of comments on merged pull requests, we found that the model taking into account only fixed effects ($R^2_m = 0.50$) fits the data well. However, there is also variability from the random effects ($R^2_c = 0.56$). We observe that *time-to-close pull requests explains the largest amount of variability in the model*, indicating that communication during the pull request review is strongly associated with the time to merge it. Regarding the bot effects, there is a discontinuity at adoption time, followed by a statistically significant decrease after the bot's introduction.

	Merged Pull Requests		Non-merged Pull Requests	
	Coefficients	SS	Coefficients	SS
Intercept	-0.096***		-0.123***	
TimeSinceFirstPullRequest	0.00000	20.0	-0.00002*	24.4
log(TotalPullRequestAuthors)	0.053***	163.6	0.069***	621.1
log(TotalCommits)	-0.014***	36.6	-0.009**	106.0
log(OpenedPullRequests)	0.079***	1002.8	0.072***	1362.9
log(TimeToClosePullRequests)	0.093***	3239.7	0.101***	4615.5
log(PullRequestCommits)	0.093***	55.0	0.123***	119.4
time	-0.001	1.0	-0.001	7.2
interventionTrue	0.023**	0.8	-0.025***	1.1
time_after_intervention	-0.002*	0.5	0.0001	0.0
Marginal R^2		0.50		0.66
Conditional R^2		0.56		0.70

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$. *SS stands for "Sum of Squares".*

Table 3.4: *The Effect of Code Review bots on Pull Request Comments. The response is* ***log(median of comments)*** *per month.*

As above, the model of non-merged pull requests fits the data well ($R^2_m = 0.66$) and there is also variability explained by the random variables ($R^2_c = 0.70$). This model also suggests that communication during the pull request review is strongly associated with the time to reject the pull request. Table 3.4 shows that the effect of bot adoption on non-merged pull requests differs from the effect on merged ones. The statistical significance of the *intervention* coefficient indicates that the adoption of code review bots slightly affected communication; however, there is no bot effect in the long run.

Since our model for merged pull requests shows a decrease in the number of comments after bot adoption, we rejected $\mathbf{H}_{2.1}$. Still, given that our model for non-merged pull requests could not observe any statistically significant bot effect as time passes, we cannot accept $\mathbf{H}_{2.2}$.

Effects in Communication. On average, there is less monthly communication on merged pull requests after adopting a code review bot. However, the monthly communication on non-merged pull requests does not change as time passes.

Effects in Pull Request Resolution Efficiency

In the exploratory case study, we found that the monthly time to close pull requests increased after bot adoption. Next, we fitted two RDD models, for both merged and non-merged pull requests, where *median of time to close pull requests* per month is the dependent variable. The results are shown in Table 3.5.

Analyzing the results of the effect of code review bots on the latency to merge pull requests, we found that combined fixed-and-random effects fit the data better than the

	Merged Pull Requests		Non-merged Pull Requests	
	Coefficients	SS	Coefficients	SS
Intercept	0.377^{**}		0.221	
TimeSinceFirstPullRequest	0.0002^{**}	452	0.00001	891
log(TotalPullRequestAuthors)	0.208^{***}	2186	0.166^{***}	21320
log(TotalCommits)	-0.145^{***}	824	-0.057^{**}	4770
log(OpenedPullRequests)	0.120^{***}	34444	0.240^{***}	50376
log(PullRequestComments)	2.472^{***}	117571	3.326^{***}	176312
log(PullRequestCommits)	2.275^{***}	47117	1.721^{***}	26733
time	0.027^{***}	3007	0.012^{**}	56
interventionTrue	0.256^{***}	128	-0.056	9
time_after_intervention	0.009	6	-0.028^{***}	66
Marginal R^2		0.61		0.69
Conditional R^2		0.67		0.72

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$. *SS stands for "Sum of Squares".*

Table 3.5: *The Effect of Code Review bots on time-to-close PRs. The response is* ***log(median of time-to-close PRs)*** *per month.*

fixed effects only ($R^2_c = 0.67$ vs $R^2_m = 0.61$). Although several variables affect the trends of pull request latency, communication during the pull requests is responsible for most of the variability in the data. This indicates the expected results: the more effort contributors expend discussing the contribution, the more time the contribution takes to merge. The number of commits also explains the amount of data variability, since a project with many changes needs more time to review and merge them. Moreover, we observe an increasing trend before adoption, followed by a statistically significant discontinuity at adoption. After adoption, however, there is no bot effect on the time to merge pull requests since the *time_after_intervention* coefficient is not statistically significant.

Turning to the model of non-merged pull requests, we note that it fits the data well ($R^2_m = 0.69$), and there is also a variability explained by the random effects ($R^2_c = 0.72$). As above, communication during the pull requests is responsible for most of the variability encountered in the results. In this model, the number of received contributions is important to explain variability in the data—projects with many contributions need more time to review and reject them. The effect of bot adoption on the time spent to reject pull requests differs from the previous model. Regarding the time series predictors, the model did not detect any discontinuity at adoption time. However, the positive trend in the latency to reject pull requests before bot adoption is reversed toward a decrease after adoption.

Thus, since we could not observe statistically significant bot effects as time passes, we cannot confirm $\mathbf{H}_{3.1}$. Further, as the model of non-merged pull requests shows a decrease in the monthly time to close pull requests, we reject $\mathbf{H}_{3.2}$.

Effects in PR Resolution Efficiency. After adopting the code review bot, on average less time is required from maintainers to review and reject pull requests. However, the time required to review and accept a pull request does not change after code review bot adoption.

Effects in Commits

Finally, we studied whether code review bot adoption affects the number of commits made before and during pull request review. Our hypothesis is that the monthly number of commits increases with the introduction of code review bots. Again, we fitted two models for merged and non-merged pull requests, where the *median of pull request commits* per month is the dependent variable. The results are shown in Table 3.6.

	Merged Pull Requests		Non-merged Pull Requests	
	Coefficients	SS	Coefficients	SS
Intercept	0.358***		0.063	
TimeSinceFirstPullRequest	0.0001***	0.30	0.00002	5.7
log(TotalPullRequestAuthors)	-0.144***	0.02	-0.058***	202.2
log(TotalCommits)	0.017***	74.04	0.028***	171.9
log(OpenedPullRequests)	0.163***	1513.60	0.125***	1502.9
log(PullRequestComments)	0.520***	2375.74	0.600***	3306.3
time	0.001	138.60	-0.003**	8.7
interventionTrue	0.137***	33.57	0.003	0.0
time_after_intervention	0.001	0.05	0.001	0.1
Marginal R^2		0.34		0.42
Conditional R^2		0.48		0.50

$^{***} p < 0.001$, $^{**} p < 0.01$, $^{*} p < 0.05$. *SS stands for "Sum of Squares".*

Table 3.6: *The Effect of Code review bots on Pull Request commits. The response is* ***log(median of Pull Request commits)*** *per month.*

Analyzing the model of commits on merged pull requests, we found that the combined fixed-and-random effects ($R^2_c = 0.48$) fit the data better than the fixed effects ($R^2_m = 0.34$), showing that most of the explained variability in the data is associated with project-to-project and language-to-language variability, rather than with the fixed effects. The statistical significance of the *intervention* coefficient indicates that the adoption of code review bots affected the number of commits only at the moment of adoption. Additionally, from Table 3.6, we can also observe that the number of pull request comments per month explains most of the variability in the result. This result suggests that the more comments there are, the more commits there will be, as discussed above.

Investigating the results of the non-merged pull request model, we found that the model fits the data well and that the random effects are again important in this regard. We also observe from Table 3.6 that the adoption of a bot is not associated with the number of commits on non-merged pull requests, since *intervention* and *time_after_intervention* coefficients are not statistically significant.

Based on models for merged and non-merged pull requests, we could not observe statistically significant effects of bot adoption. Therefore, we cannot confirm both $\mathbf{H}_{4.1}$ and $\mathbf{H}_{4.2}$.

Effects in Commits. After adopting a code review bot, the monthly trend in the median of pull request commits does not change for both merged and non-merged pull requests.

3.3.2 Developers' Perspective on the Reasons for the Observed Effects (RQ2)

As explained in Section 3.2.3, we presented to open-source developers the main findings of our statistical approach: "*After adopting a code review bot there are more merged pull requests, less communication between developers, fewer rejected pull requests, and faster rejections.*" We asked them to conjecture on the possible reasons for each of these results.

We grouped the participants' answers into 5 categories, as can be seen in Table 3.7. We associate one of the effects with its correspondent reasons whenever participants explicitly mentioned this relationship. We also added a mark (✓) to highlight which effects are explained by each one of the reasons, according to the participants' responses.

Reason	**#**	**Explains**			
		More Merged PRs	**Fewer Comments**	**Fewer Rejected PRs**	**Faster Rejections**
More visibility and transparency of the contribution state	8	✓	✓	✓	✓
More confidence in the process in place	8	✓	✓		✓
Bot feedback changes developers' discussion focus	8		✓		
Bot feedback pushes contributors to take an action	5	✓		✓	
Bot feedback perceived as noise	2		✓		

Table 3.7: *Main reasons for the findings from the RDD models.*

More visibility and transparency of the contribution state. Most of the participants claimed that when a project has bots that provide detailed information on code quality metrics, especially in the sense of coverage metrics, both maintainers and contributors can more quickly gain a general idea of the quality of the contributions. As stated by P6: "*bots are able to raise visibility, both for the contributor and for the maintainer. They can make it more clear more quickly the state of that contribution.*" More than obtaining clarity on the quality of the code, it is also easy for maintainers to verify whether the pull request contributors will improve their contribution toward achieving acceptance. Thus, they conjecture that all bot effects we found during the statistical analysis might be explained by this enhanced feedback given by the bot.

As soon as contributors submit their pull requests, the code review bot posts a detailed comment regarding the code coverage. In the P7 experience, the "*immediate feedback of the quality of [code coverage] on the pull request*" is closely related to the increasing acceptance rate. If the pull request does not affect code coverage in a negative way, then maintainers are able to "*much more quickly judge whether or not it's a reasonable request*" (P4). On the other side of the spectrum, if the pull requests fail the tests and decreases the coverage, then the maintainers "*will not bother with that pull request at all, and just reject it*" (P4). Also maintainers "*are more inclined to directly reject the pull request*" since it does not respect the rules imposed by the project. In some cases, maintainers expect that the contributor will take an action based on the bot comments, as explained by P6: "*if [contributors] are not following up and resolving the issue, it makes it more clear to the maintainer that it's not an acceptable contribution.*"

Participants also recognize that these bots are usually "*pretty good at explaining very precisely*" (P2) and not merely stating that "*[maintainers] will not accept the pull request*" (P2) without further explanation. For example, if the coverage decreased, the bot will post "*your pull request dropped the test coverage from 95 to 94%. And these are the lines you edit that are not covered. So, please add tests to cover these specific lines.*" (P2), which according to P2 is extremely useful for a contributor. According to P1, for example, the visibility of the bot comments helps maintainers to make sure contributors understand why the pull request has been rejected without the necessity of engaging in a long discussion: "*now the maintainer can just point at it and be like 'you didn't pass the status check, because you didn't write tests.' It is more obvious*".

More confidence in the process in place. According to the participants, one of the reasons for more pull requests being merged after the code review bot introduction is that these bots act as quality gatekeepers. For example, P1 mentioned that *"by having other metrics, like code coverage, to be able to say 'Great! I know that at least a test has been written for that line of code', there is some sort of gatekeeping."* Besides the effect of merging more pull requests, participants also mentioned another effect: "*accepting code contributions can be much, much faster*" (P2). Basically, code review bots are used as a way to achieve "*automatic verification*" (P7). According to P7, if the bots confirm that the change is correct, then "*the developer is more convinced that the change is useful and valid.*" In the opposite way, if the bots shows that the change is incorrect, the pull request will be rejected faster, as it does not require "*human interaction to arrive at this conclusion*" (P7), which implies less communication between developers. Furthermore, P4 also relates the confidence in the bot as one of the reasons for less communication between developers: *the fact that there is less communication between the contributors and maintainers might be an effect that we can get a bit overdependent on bots, in the sense you trust them too much.*" Therefore, since maintainers trust the bots' feedback, they "*ask fewer questions*" (P2).

Bot feedback changes developers' discussion focus. Participants recurrently mentioned that bot comments enabled them to focus on other high-priority discussions, which led to a decrease in the communication between the project maintainers and contributors on pull requests. To some extent this decrease occurs since *it's not necessary anymore,* *because a lot of that [comments] are [already] handled automatically by the bots*" (P4). In P3's experience, maintainers "*talk more for new developers, to text them usually things like 'Add new test please' and then [maintainers] don't have to [make] that kind of comment[] anymore. That's why there's less communication.*"

Moreover, when receiving non-human feedback, contributors are less likely to start a broader discussion about the viability or necessity of software testing, as explained by P2: "*once you have set up the bots, and it is automated, people are less likely to argue about it, which is just a nice effect of bots. Especially for bots that kind of point out failures. I think it's good to have that from bots, and not from people.*" There are some exceptions, however, when contributors experienced an increase in communication incurred by the bot comments, especially when they do not understand how they might increase the coverage rate. As posed by P9: *In my experience, it causes a longer discussion, because then I have to talk to the engineers like 'hey, what kind of a test should I add such as coveralls passes?"'*

Bot feedback pushes contributors to take an action. Also related to the transparency introduced by the bot comments, and in line with the idea of code review bots as quality gatekeepers, these bots lead developers to take an action: "*It gives me clear instructions on what I have to do to resolve it. So, I'm very likely to act on it*" (P2). These bots protect developers from reducing the code's coverage. Therefore, developers would consider either closing the pull request, if it is not worth their time, or following up with the necessary changes: *you have this systematic check that says 'okay, that's not good.' And then the developer is saying, 'okay, it won't be accepted if I don't provide the test'* "(P3).

Bot feedback perceived as noise. Although less recurrent, participants mentioned that in some cases bot comments might be perceived as noise by developers, which disrupts the conversation in the pull request. On the one hand, "*comments from code coverage bots tend to give you more visibility and provide more context[]*" (P6). On the other hand, developers complain about the noise these comments introduce to the communication channel. According to P7, the repetitive comments of code coverage bots are "*disrupting the conversation*", since "*if you have to develop a certain conversation and you have a bot message, this could have a negative impact on the conversation.*" One of the consequences of this noise incurred by the repetitive bot comments is that "*[developers] pay less attention to it*" (P7), impacting the developers communication.

We also asked developers whether they have seen the observed effects on their own projects, and what are the other effects they attribute to the code review bot adoption. The most recurrent (8) observed effect was less communication. As stated by P10: "*I remember one of the maintainers saying 'the tests are missing here.' She always had to post that comment. Then, we adopted the bot to comment on the coverage and had no need for her to comment anymore.*" Also, 6 participants observed fewer pull requests rejections and faster rejections, and 5 participants have observed more merged pull requests. Finally, developers did not attribute any other effect to the bot introduction.

> **Summary of reasons.** Project maintainers and contributors reported several reasons for more merged pull requests, fewer comments, and fewer and faster rejections.

According to them, bot comments help them to understand the state and quality of the contribution, making maintainers more confident to merge pull requests, which also changes the focus of developer discussions.

3.4 Limitations and Threats to Validity

In this section, we discuss the limitations and potential threats to validity of our study, their potential impact on the results, and how we have mitigated them (Wohlin *et al.*, 2012).

External Validity: While our results only apply to OSS projects hosted on GitHub, many relevant projects are currently hosted on this platform (Dias *et al.*, 2016). Our selection of projects also limits our results. Therefore, even though we considered a large number of projects and our results indicate general trends, we recommend running segmented analyses when applying our results to a given project. For replication purposes, we made our data and source code publicly available.[4]

Construct Validity: One of the constructs in our study is the "first bot comment on a pull request" as a proxy to the "time of bot adoption" on a project. A more precise definition of this adoption time would have involved the integration date, which is not provided by the GitHub API. Hence, the validity of the "time of bot adoption" construct might have been threatened by the definition. We reduce this threat by excluding the period of 15 days immediately before and after adoption from all analyses. Moreover, Kalliamvakou et al. (Kalliamvakou *et al.*, 2014) stated that many merged pull requests appear non-merged, which could also affect the construct validity of our study, since we consider the number of merged pull requests. To increase construct validity and improve the reliability of our qualitative findings, we employed a constant comparison method (Glaser and Anselm L Strauss, 2017). In this method, each interpretation is constantly compared with existing findings as it emerges from the qualitative analysis.

Internal Validity: To reduce internal threats, we applied multiple data filtering steps to the statistical models. To confirm the robustness of our models, we varied the data filtering criteria, for example, by filtering projects that did not receive pull requests in all months, instead of at least 6 months, and observed similar phenomena. Projects that disabled the bot during the period we considered might be a threat. However, detecting whether a project disabled the bot or not is challenging. The GitHub API does not provide this information. We reduce this threat by removing from our dataset projects without bot comments during the last four months of analysis. Additionally, we added several controls that might influence the independent variables to reduce confounding factors. However, in addition to the already identified dependent variables, there might be other factors that influence the activities related to pull requests. These factors could include the adoption of other code review bots, or even other types of bots and non-bot automation. To treat this, we removed projects that adopted more than one bot, based on the list of bots provided by Wessel et al. (Wessel, Souza, *et al.*, 2018). To ensure information

[4] https://doi.org/10.5281/zenodo.4618498

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 (A. Strauss and Juliet M Corbin, 1997), sampling should be discontinued once the collected data is considered sufficiently dense and data collection no longer generates new information.

3.5 Discussion

Adding a code review bot to a project often represents the desire to enhance feedback about the contributions, helping contributors and maintainers, and achieving improved interpersonal communication, as already discussed by Storey and Zagalsky (Storey and Zagalsky, 2016). Additionally, code review bots can guide contributors toward detecting change effects before maintainers triage the pull requests (Wessel, Souza, *et al.*, 2018), ensuring high-quality standards. In this Chapter, we focused on monthly activity indicators that are not primarily related to bot adoption, but might be impacted by it. We found that the bot adoption has a statistically significant effect on a variety of activity indicators.

According to the regression results, the monthly number of merged pull requests increased, even faster, after the code review bot adoption. In addition, the number of non-merged pull requests continued to decrease, even faster, after bot adoption. These models showed that after adopting the bot, maintainers started to deal with an increasing influx of contributions ready to be further reviewed and integrated into the codebase. Also, these findings confirm the hypothesis we formulated based on the exploratory case study. According to our participants, the increase in the monthly number of merged pull requests, as well as the decrease in the monthly number of non-merged one, are explained by the transparency introduced by the bot feedback. Contributors started to have faster and clearer feedback on what they needed to do to have their contribution accepted. Further, participants also mentioned that contributors have been pushed to enhance their pull requests based on bot feedback.

In addition, we noticed that just after the adoption of the code review bot the median number of comments slightly increased for merged pull requests. The number of comments on these pull requests could increase due to contributions that drastically reduced the coverage, stimulating discussions between maintainers and contributors. This can happen especially at the beginning of bot adoption, since contributors might be unfamiliar with bot feedback. After that initial period, we found that the median number of comments on merged pull requests decreased each month. According to our participants, less communication could be explained by the transparency and confidence developers gain from bot feedback. Also, developers mentioned that after bot adoption the focus of the developers discussion changed, since there is no need for certain discussions related to coverage. Considering non-merged pull requests, there is no significant change in the number of comments as time passes. These results differ from the case study results, indicating that individual projects reveal different results, which are likely caused by other project-specific characteristics.

From the regression results, we also noticed an increase in the time spent to merge pull requests just after bot adoption. It makes sense from the contributors' side, since the bot introduces a secondary evaluation step. Especially at the beginning of the adoption, the code review bot might increase the time to merge pull requests due to the need to learn how to meet all bot requirements and obtain a stable code. Maintainers might also deal with an increase in the volume of contributions ready to review and merge, impacting the time spent to review all of them. Further, the regression model shows a decrease in the time spent to review and reject pull requests. Overall, according with our participants it indicates that after the bot adoption maintainers stopped expending effort on pull requests that were not likely to be integrated into the codebase.

As we found in the model of commits on merged pull requests, just after the adoption of the bot the median number of pull request commits increased. The bot provides immediate feedback in terms of proof of failure, which can lead contributors to submit code modifications to change the bot feedback and have their contribution accepted. Overall, the regression models reveal that the monthly number of commits did not change for both merged and non-merged pull requests as time passed. These results differ from the case study results. Nevertheless, even if there is an increase in the number of commits reported in the case study, overall the monthly number of commits are quite stable. For example, for CakePHP it varies from 1 to 2 for merged pull requests, and 1 to 4 for non-merged pull requests. Additionally, in the main study, we account for control variables, rather than analyzing the monthly number of commits interdependently. As presented in Section 3.3.1, for example, the number of comments on pull requests explains the largest amount of variability in these models, indicating that the number of commits is strongly associated with the communication during the pull request review.

3.5.1 Implications and Future Work

In the following, we discuss implications and future work for researchers and practitioners in light of our results and related literature.

Implications for Project Members

Projects need to make informed decisions on whether to adopt code review bots (or software bots in general) into their projects and how to use them effectively. We found that the dynamics of pull requests changed following the adoption of code review bots. Hence, besides understanding the effects on code quality, practitioners and open-source developers can leverage our results to be aware of other consequences of bot adoption and take countermeasures to avoid the undesired effects.

Implications for Newcomers

Our previous work reported that although bots could make it easier for some newcomers to submit a high-quality pull request, bots can also provide newcomers with information that can lead to rework, discussion, and ultimately dropping out from contributing (Wessel, Serebrenik, I. Wiese, I. Steinmacher, and Marco Aurelio Gerosa, 2020). It is reasonable to expect that newcomers who receive friendly feedback will have a higher engagement level and thus sustain their participation on the project. Hence, future research can help bot designers by providing guidelines and insights on how to support new contributors.

Additional effort is also necessary to investigate the impact of code review bots' feedback for newcomers, who already face a variety of barriers (Balali *et al.*, 2018; I. Steinmacher, T. Conte, *et al.*, 2015).

Implications for Researchers

For researchers interested in software bots, it is important to understand the role of code review bots in the bot landscape. It is important to understand how such bots affect the interplay of developers in their effort to develop software, and our study provides the first step in this direction. Considering that bot output is mostly text-based, how bots present content can highly impact developers' perceptions (Liu *et al.*, 2020; Chaves and Marco Aurelio Gerosa, 2020). Additional effort is necessary to investigate how the developers' cognitive styles (Vorvoreanu *et al.*, 2019; Mendez *et al.*, 2018) might influence the way 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. Other social characteristics of the bots can also be investigated in this context (Chaves and Marco Aurelio Gerosa, 2020). Future research can lead to a set of guidelines on how to design effective messages for different cognitive styles and developer profiles. Further, developers complain about the information overload caused by repetitive bot behavior on pull requests, which has received some attention from the research community (Erlenhov, Neto, *et al.*, 2016), but remains a challenging problem. In fact, there is room for improvement on human-bot collaboration on social coding platforms. When they are overloaded with information, teams must adapt and change their communication behavior (Ellwart *et al.*, 2015). Therefore, there is also an opportunity to investigate changes in developers' behavior imposed by the effects of information overload. Additional research can also investigate how to use code reviews bots to support the training of new software engineers (Pinto *et al.*, 2017).

3.6 Final Considerations

In this chapter, we presented an exploratory empirical investigation of the effects of adopting bots to support the code review process on pull requests. While several code review bots have been proposed and adopted by the OSS community, relatively little has been done to evaluate the state of practice. To understand the impact on practice, we statistically analyzed data from $1,194$ open source projects hosted on GitHub. Further, we had an in-depth investigation into the reasons of the identified impacts. We interviewed 12 project maintainers and contributors experienced with code review bots.

By modeling the data around the introduction of a code review bot, we notice different results from merged pull requests and non-merged ones. We see that the monthly number of merged pull requests of a project increases after the adoption of a code review bot, requiring less communication between maintainers and contributors. At the same time, code review bots can lead projects to reject fewer pull requests. Afterwards, when interviewing developers we found a comprehensive set of reasons for these effects. First of all, bot comments help contributors and maintainers to be aware the state and quality of the contribution, making maintainers more confident to merge pull requests, which also changes the focus of developers' discussions.

In the next chapter, we delve into the challenges caused by the interaction of bots on pull requests.

Chapter 4

Challenges of Interacting with Software Bots

This chapter reports our work on identifying the challenges incurred by the bots interaction on pull requests. 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. The study described in this chapter was published at CSCW 2021 (Wessel, I. Wiese, *et al.*, 2021).

4.1 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 4.1 shows an overview of the research design employed in this study.

4.1.1 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

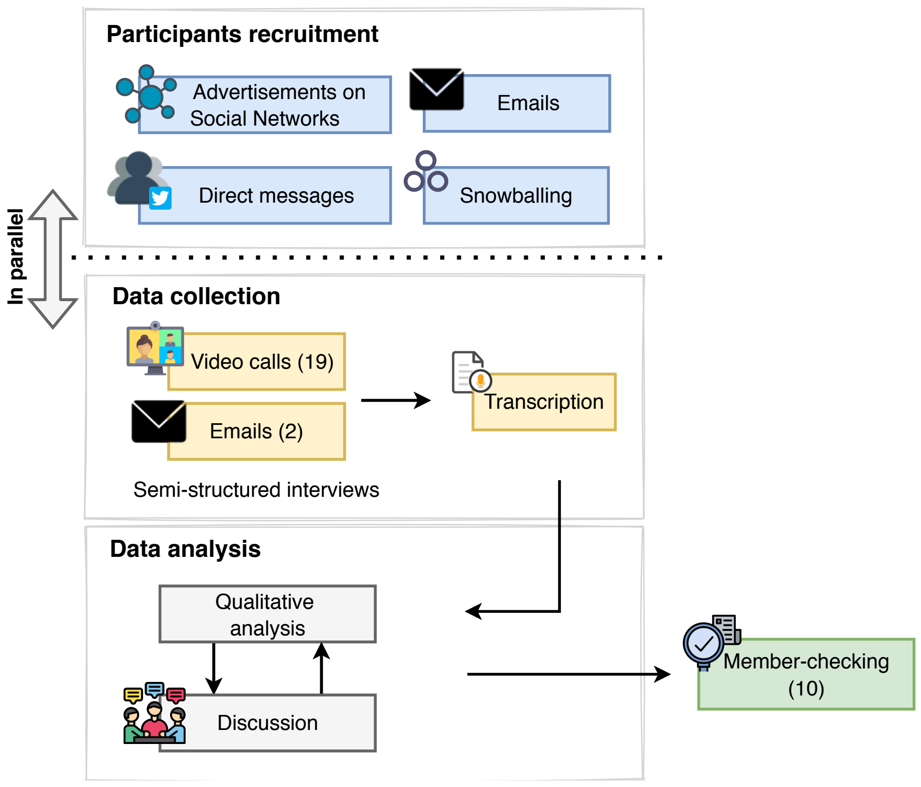


Figure 4.1: *Research Design Overview*

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.

As a result of our recruitment, we interviewed 21 participants—identified here as P1 – P21. Table 4.1 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

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*

Table 4.1: *Demographics of interviewees*

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.

4.1.2 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 (Hove and Anda, 2005). 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[1]. Each interview was conducted remotely by the author of this dissertation and lasted, on average, 46 minutes.

4.1.3 Data analysis

We qualitatively analyzed the interview transcripts, performing open and axial coding procedures (A. L. Strauss and J. M. Corbin, 1998; Stol *et al.*, 2016) 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 author of this dissertation conducted a preliminary analysis, identifying the main codes. Then, the author discussed the coding in weekly hands-on meetings with three other experienced researchers, aiming to increase the reliability of the results and mitigate bias (A. L. Strauss and J. M. Corbin, 1998; Patton, 2014). In these meetings, all the researchers revisited codes, definitions, and their relationships until reaching an agreement. Afterwards, the 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 (Glaser and Anselm L Strauss, 2017), 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[1]. The code book includes the code names, descriptions, and examples of quotes for all categories.

4.1.4 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 (Merriam, 1998). According to Charmaz (2006), 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*".

4.2 Results

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.

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

4.2.1 Challenges incurred by bots

The interviewees reported social and technical challenges related to the development, adoption, and interaction of bots on pull requests. Figure 4.2 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 Chapter 2. In summary, we 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).

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 4.2.2, 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*

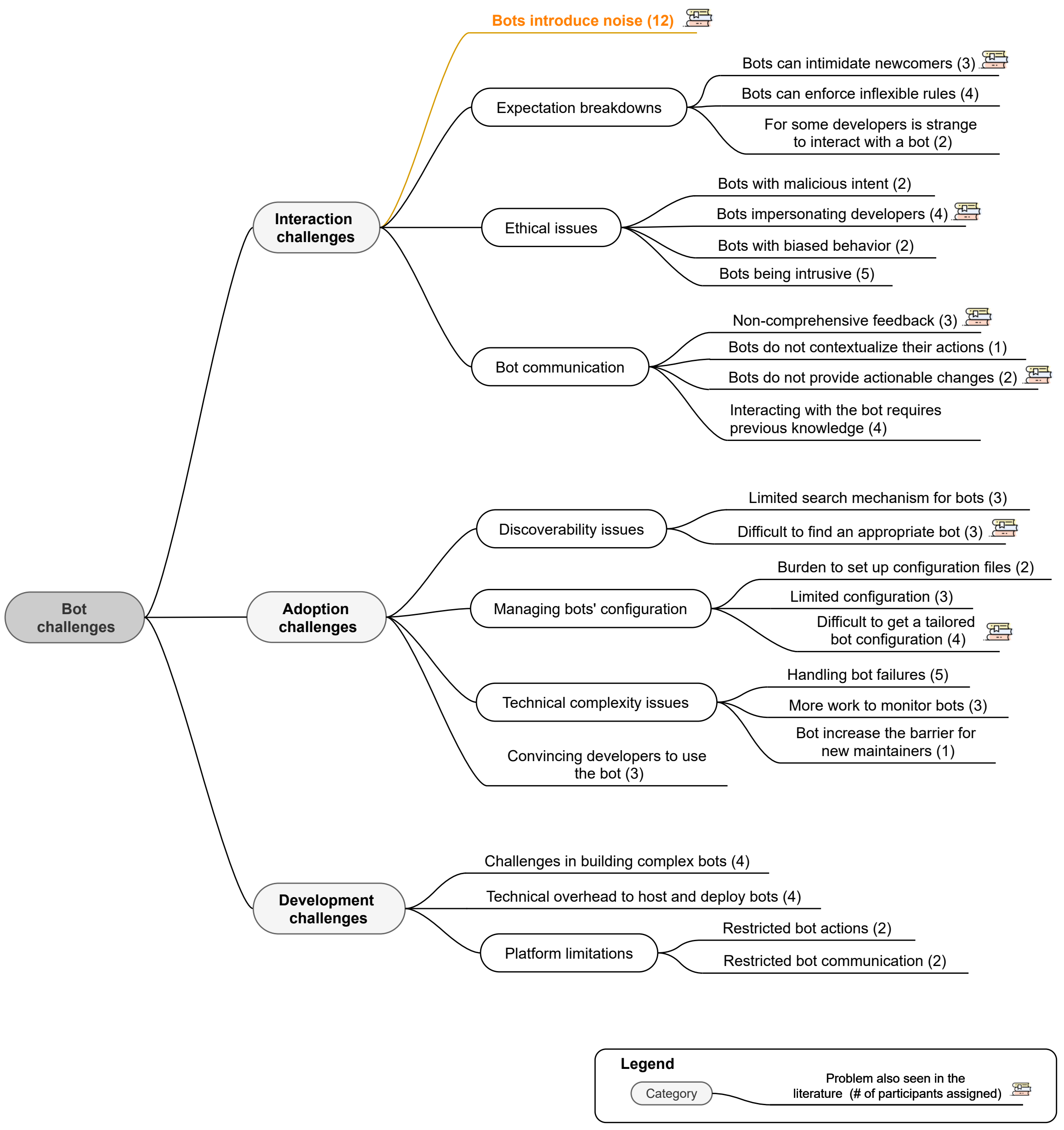

Bots introduce noise (12)
Bots can intimidate newcomers (3)
Bots can enforce inflexible rules (4)
Expectation breakdowns
For some developers is strange to interact with a bot (2)
Bots with malicious intent (2)
Interaction challenges
Bots impersonating developers (4)
Ethical issues
Bots with biased behavior (2)
Bots being intrusive (5)
Non-comprehensive feedback (3)
Bots do not contextualize their actions (1)
Bot communication
Bots do not provide actionable changes (2)
Interacting with the bot requires previous knowledge (4)
Limited search mechanism for bots (3)
Discoverability issues
Difficult to find an appropriate bot (3)
Burden to set up configuration files (2)
Limited configuration (3)
Bot challenges
Adoption challenges
Managing bots' configuration
Difficult to get a tailored bot configuration (4)
Handling bot failures (5)
Technical complexity issues
More work to monitor bots (3)
Bot increase the barrier for new maintainers (1)
Convincing developers to use the bot (3)
Challenges in building complex bots (4)
Technical overhead to host and deploy bots (4)
Development challenges
Restricted bot actions (2)
Platform limitations
Restricted bot communication (2)
Legend
Category
Problem also seen in the literature (# of participants assigned)

Figure 4.2: *Hierarchical Categorization of Bot Challenges*

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.[2] 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 vulnerability into the source code by merging a pull request. Further, according to P9, 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).

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 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**

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

(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.

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.

4.2.2 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 4.3 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.

In the following, we present in detail the theory about noise introduced by bots described in Figure 4.4. 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).

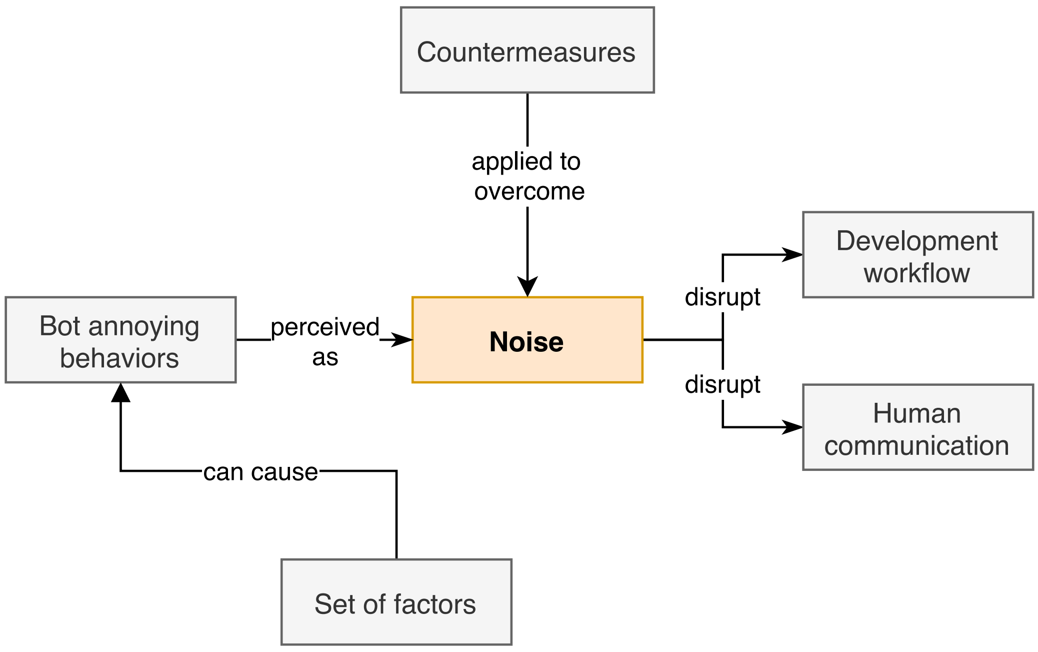


Figure 4.3: *High-level concepts and relationships of Bots' Noise Theory*

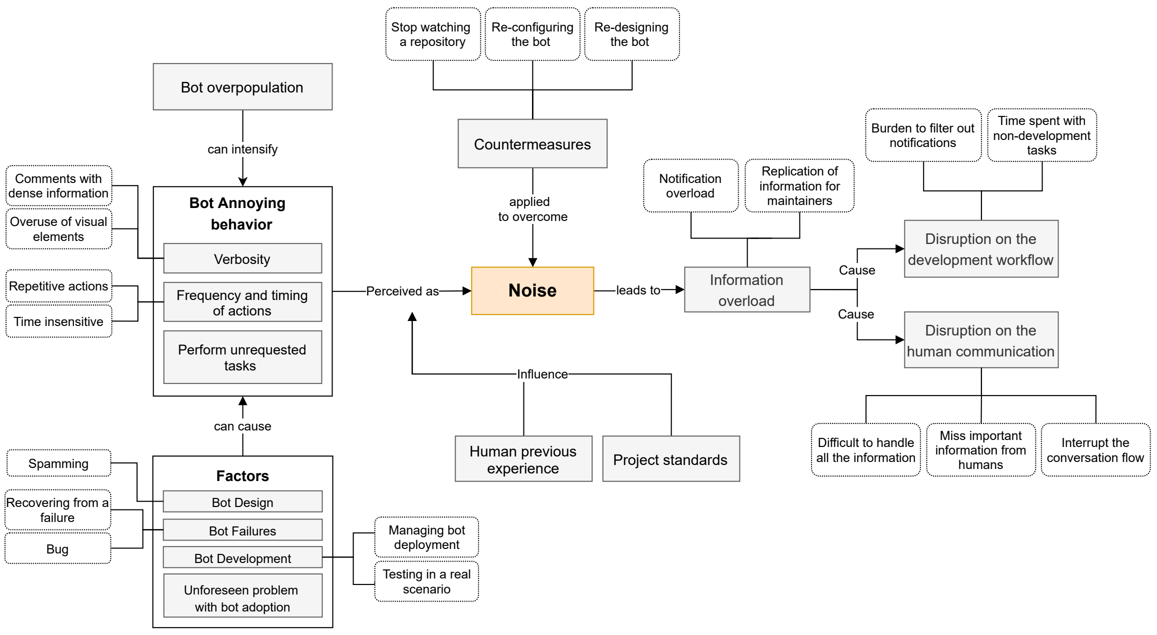

Stop watching a repository
Re-configuring the bot
Re-designing the bot
Countermeasures
applied to overcome
Bot overpopulation
can intensify
Bot Annoying behavior
Verbosity
Frequency and timing of actions
Perform unrequested tasks
Comments with dense information
Overuse of visual elements
Repetitive actions
Time insensitive
Perceived as
Noise
leads to
Information overload
Notification overload
Replication of information for maintainers
Cause
Cause
Disruption on the development workflow
Burden to filter out notifications
Time spent with non-development tasks
Disruption on the human communication
Difficult to handle all the information
Miss important information from humans
Interrupt the conversation flow
Influence
Human previous experience
Project standards
can cause
Factors
Bot Design
Bot Failures
Bot Development
Unforeseen problem with bot adoption
Spamming
Recovering from a failure
Bug
Managing bot deployment
Testing in a real scenario

Figure 4.4: *Theory about noise introduced by bots*

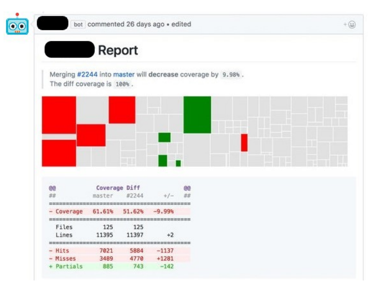
Bot annoying behaviors

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 4.5a 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 4.5b, 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 4.5c depicts a bot spamming a repository with an unsolicited pull request.



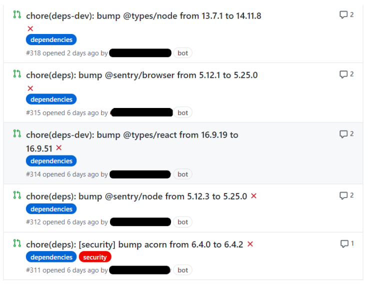
[redacted] bot commented 26 days ago • edited

[redacted] Report

Merging #2244 into master will decrease coverage by 9.98%.
The diff coverage is 100%.

```
@@            Coverage Diff            @@
##           master    #2244     +/-   ##
==========================================
- Coverage   61.61%   51.62%   -9.99%
==========================================
  Files         125      125
  Lines       11395    11397       +2
==========================================
- Hits         7021     5884    -1137
- Misses       3489     4770    +1281
+ Partials      885      743     -142
```

(a) *Verbosity*



chore(deps-dev): bump @types/node from 13.7.1 to 14.11.8 ✗ 2
dependencies
#318 opened 2 days ago by [redacted] bot

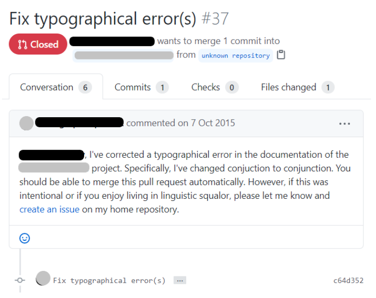
chore(deps): bump @sentry/browser from 5.12.1 to 5.25.0 ✗ 2
dependencies
#315 opened 6 days ago by [redacted] bot

chore(deps-dev): bump @types/react from 16.9.19 to 16.9.51 ✗ 2
dependencies
#314 opened 6 days ago by [redacted] bot

chore(deps): bump @sentry/node from 5.12.3 to 5.25.0 ✗ 2
dependencies
#312 opened 6 days ago by [redacted] bot

chore(deps): [security] bump acorn from 6.4.0 to 6.4.2 ✗ 1
dependencies security
#311 opened 6 days ago by [redacted] bot

(b) *High frequent actions*



Fix typographical error(s) #37

Closed [redacted] wants to merge 1 commit into from unknown repository

Conversation 6 Commits 1 Checks 0 Files changed 1

[redacted] commented on 7 Oct 2015

[redacted], I've corrected a typographical error in the documentation of the project. Specifically, I've changed conjuction to conjunction. You should be able to merge this pull request automatically. However, if this was intentional or if you enjoy living in linguistic squalor, please let me know and create an issue on my home repository.

Fix typographical error(s) c64d352

(c) *Unsolicited actions*

Figure 4.5: *Examples of annoying behaviors from the state-of-the-practice*

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 4.2.1. 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 4.2.1, 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 4.2.1), 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.*

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.*"

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.*"

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, Yoo, *et al.* (2018) and Peng and Ma (2019) 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.

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.

4.3 Discussion

In this section, we discuss our main findings, comparing them with the state-of-the-art.

Bots on GitHub serve as an interface to integrate humans and services (Wessel, Souza, *et al.*, 2018; Storey and Zagalsky, 2016). 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 (2016) and Paikari and Hoek (2018), 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 (Storey, Serebrenik, *et al.*, 2020). 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 C. Lebeuf, Storey, *et al.* (2018), 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 (Liu *et al.*, 2020). 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 (Wessel, Souza, *et al.*, 2018). Brown and Parnin (2019) 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, Souza, *et al.* (2018) has already mentioned that support for newcomers is both challenging and desirable. In a subsequent study, Wessel, Serebrenik, I. Wiese, I. Steinmacher, and Marco Aurelio Gerosa (2020) 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 (Vorvoreanu *et al.*, 2019; Mendez *et al.*, 2018) 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 (I. Steinmacher, Chaves, *et al.*, 2013). 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 (Storey, Serebrenik, *et al.*, 2020). For bots on Wikipedia, for example, this behavior is expressly prohibited (Müller-Birn *et al.*, 2013). At the same time, Murgia *et al.* (2016) 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.* (2021) 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 (Shannon, 2001). 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, Neto, *et al.* (2016), 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 (1956) 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 (Brown and Parnin, 2019; Mirhosseini and Parnin, 2017; Peng, Yoo, *et al.*, 2018; Peng and Ma, 2019). According to Erlenhov, Neto, *et al.* (2016), 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.* (2018) 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 (Simone *et al.*, 1995; López and Guerrero, 2016), which have not been yet explored by social coding platforms. As pointed out by Iqbal and Horvitz (2010), users want to be notified, but they also want to have ways to filter notifications and determine how they will be notified. I. Steinmacher, Chaves, *et al.* (2013) 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 (Bawden and Robinson, 2009; Jones *et al.*, 2004; Nematzadeh *et al.*, 2016). According to Nematzadeh *et al.* (2016), 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.* (2016) can impact the overall engagement of contributors and maintainers when discussing pull requests. Further, Jones *et al.* (2004) proposed a theoretical model on the impact on message dynamics of individual strategies to cope with the information overload. According to Jones *et al.* (2004), as the information overload grows, users tend to focus on and respond to simpler information, and eventually cease active participation.

4.3.1 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 (Liu *et al.*, 2020; Storey, Serebrenik, *et al.*, 2020). 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 (Gnewuch *et al.*, 2017).

Implications for Social Coding Platforms: Because of the growing use of bots for collaborative development activities (Erlenhov, Oliveira Neto, *et al.*, 2019), a proliferation of bots to automate software development tasks was expected. Recently, GitHub introduced GitHub Actions[3], 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 (Kinsman *et al.*, 2021), 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 (Müller-Birn *et al.*, 2013). Therefore, it is

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

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 (Iqbal and Horvitz, 2010), 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 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 (Liu *et al.*, 2020). 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 (Wessel, Souza, *et al.*, 2018; Wessel and I. Steinmacher, 2020; Erlenhov, Neto, *et al.*, 2016), 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 (Ellwart *et al.*, 2015). Therefore, there is also an opportunity to investigate changes in developers' behavior imposed by the effects of information overload.

4.4 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 (A. Strauss and Juliet M Corbin, 1997), 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 (Glaser and Anselm L Strauss, 2017). 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.

4.5 Final Considerations

In this chapter, 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.

Next, we use the findings from noise theory to ground the creation and evaluation of design strategies to mitigate the information overload generated by annoying bot behaviors. In the following chapter, we present the concept and preliminary evaluation of a meta-bot to mediate the action of other bots on pull requests.

Chapter 5

Designing Strategies to Mitigate Noise

In the study presented in Chapter 4, 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. 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 (Sadeddin *et al.*, 2007; Dagli, 2019; Candello, Vasconcelos, *et al.*, 2017). 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 overload caused by bots on pull requests by means of a meta-bot. Specifically, in this chapter we investigate the following research question:

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

To answer this research question, we conducted an empirical study to assess a preliminary version of the meta-bot, which aggregates the outcomes of other bots on a pull request. Afterwards, we applied Design Fiction (Blythe, 2014) as a participatory design method. This technique is used to probe, explore, and critique future technologies (Blythe, 2014). 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 a fictional 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 the fictional story, raising concerns about the use of bots and discussing the design strategies for the meta-bot.

5.1 Preliminary Conception and Evaluation of the Meta-bot

The main goal of this preliminary investigation is to compare the way meta-bot display the information (i.e. grouping and categorizing the bots' outputs) with the existing approach (i.e. multiple bots commenting on the pull request). To achieve our goal, we first implemented the meta-bot. Afterwards, we performed a user study to investigate the developers' perception and understanding of the bots' information with and without the meta-bot acting as a mediator. The results presented in the following sections resulted from a graduation thesis advised by the author of this dissertation.

5.1.1 Research Design

We implemented and evaluated a meta-bot that aggregates information from various bots on a pull request. Next, we describe the concept, preliminary implementation, and evaluation of the meta-bot.

Meta-bot concept

The meta-bot concept was inspired by Sadeddin *et al.* (2007) work. In order to deal with several responses from different bots, Sadeddin *et al.* (2007) showed that a meta-bot would obtain product information from several shopping bots, and then summarize and present it to the user. The concept of meta-bot is also present in the literature of software agents. Generalist agents are also referred to as Super Bots or meta-bots (Dagli, 2019). This is because they often combine multiple tasks and functionalities of specialist agents into a single agent. We hypothesize that *a meta-bot can mitigate the information overload created by other bots around pull requests.*

Essentially, the meta-bot way to solve interaction problems is by mediating the action of other bots used on pull requests. It would operate as a mediator between human developers and the bots in a repository. Different from other *GitHub bots*, the meta-bot would not handle specific tasks on pull requests. Instead, the meta-bot would provide additional value to the interaction of already existing bots. Moreover, once the meta-bot is integrated into a GitHub repository, it would be aware of the task-oriented bots adopted to handle the pull requests. By providing a centralized control, meta-bot would be capable of integrating and orchestrating those bots.

A way to mitigate communication noise is by restricting inconvenient bots to interact directly on pull requests. As the meaning of inconvenience can vary from project to project, the meta-bot would provide an interface that each project can previously configure bots' restrictions. Therefore, once a new pull request is opened, the meta-bot would handle bots' response based on the restrictions by aggregating the response of bots allowed to interact on that pull request.

Meta-bot implementation

We implemented the meta-bot using Probot,[1] a framework for building applications for GitHub in Node.js. We used it to facilitate the integration between our meta-bot and GitHub. The meta-bot functionality consists of consuming two types of events, "pull request opened" and "comment created." Figure 5.1 shows an overview of the meta-bot implementation.

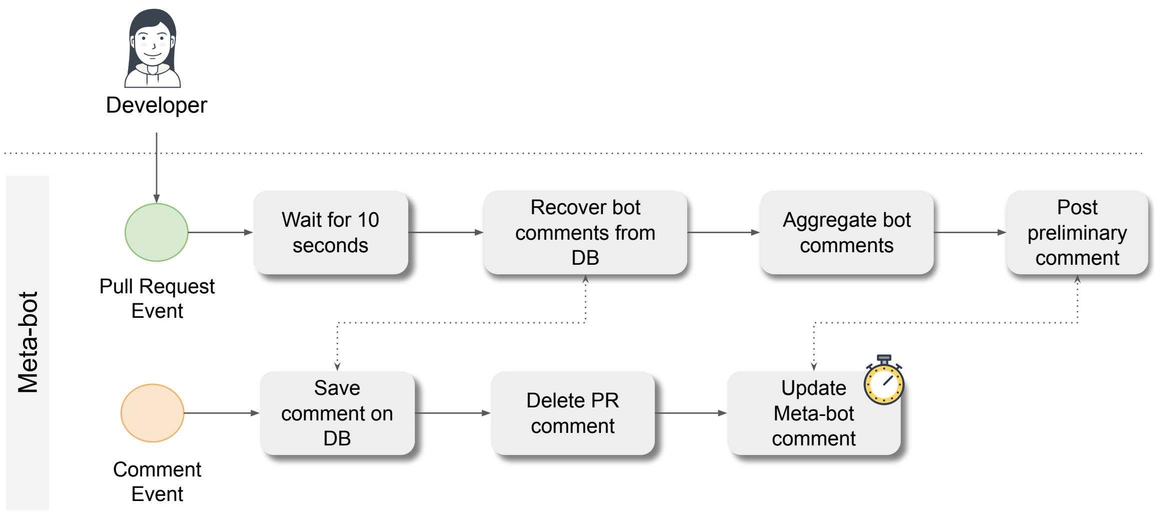


Figure 5.1: *An overview of the meta-bot implementation*

Due to several technical limitation imposed by the GitHub platform and the GitHub API, we used an strategy of deleting the bot comment once they post it in the pull request. As far as we know, the only way of preventing a bot from commenting directly in the GitHub is by modifying its source code. However, there is also a limited amount of open-source bots. Considering this, once a pull request is opened, the bots installed in the repository perform their tasks and create a comment with their results. The meta-bot then performs two actions: i) saves the bot comments in a database; ii) deletes the original comment. Before deleting a comment, the meta-bot needs to check whether the event originates from a bot or a human user. If it is from a bot, the comment is deleted and saved in the database. After its deletion, GitHub introduces the message: "metabot deleted a comment from bot-name" in the pull request timeline. This message exists to maintain traceability of the actions that occurred over the pull request's life. However, they contribute to information overload, making it an obstacle to the meta-bot's proposal. We developed an ad-hoc solution using the Violentmonkey[2] browser extensions to hide this message throughout our experiment.

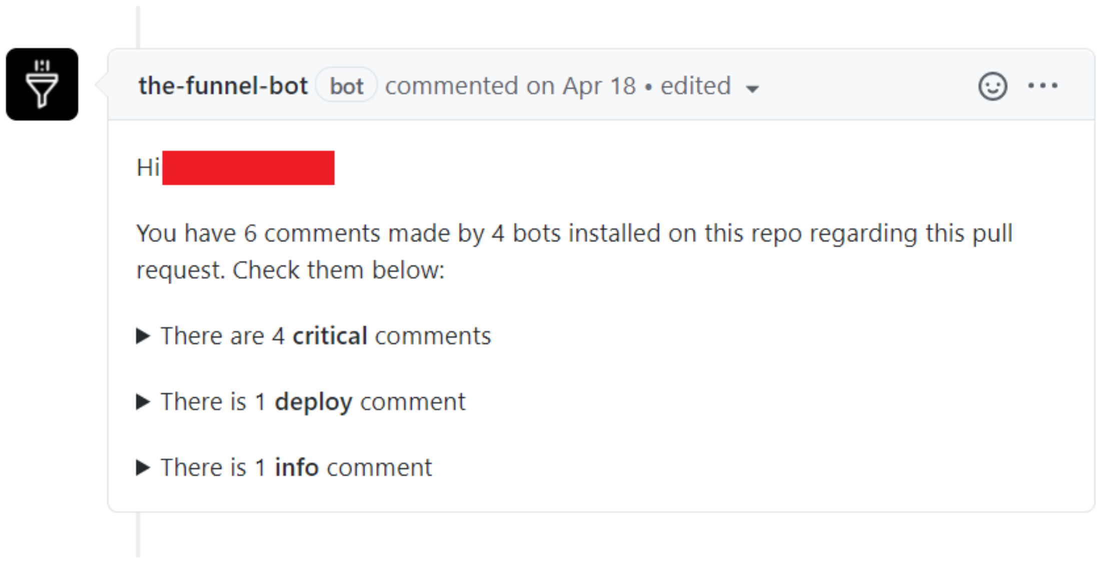
If there is already a comment from the meta-bot in this pull request, three additional actions occur: iii) retrieves all comments saved in the database; iv) aggregates the retrieved comments; v) updates the existing partial comment with the new additions. The meta-bot waits ten seconds, retrieves all comments saved in the database and aggregates them in a single comment. This way, a partial comment is created with the content coming from the

[1] https://probot.github.io/docs/

[2] https://violentmonkey.github.io/

bots that commented in the ten seconds window. Finally, the meta-bot posts the comment in the GitHub pull request and updates it as more bots finish their processing. Updating the existing meta-bot comment is necessary because not all bots installed in the repository can comment within the same time window.

The information in the meta-bot's comment is categorized by labels created according to the user's preferences. The developer can change or create them by modifying the repository configuration file according to their preferences. The partial comment turns into a complete comment when all bots comment, as shown in Figure 5.2.



Hi [redacted]

You have 6 comments made by 4 bots installed on this repo regarding this pull request. Check them below:

▶ There are 4 **critical** comments

▶ There is 1 **deploy** comment

▶ There is 1 **info** comment

Figure 5.2: *Meta-bot's complete comment in a pull request*

Meta-bot evaluation

To assess the meta-bot in a real-world scenario, we rely on Reakit[3] repository. Reakit is an open-source library used to create React applications with a focus on accessibility. This project uses four bots to support the pull request review process. Each of them is responsible for a different task; however, all bots report their outputs using comments on the pull request. The bots supporting the project's pull request are as follows:

- **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 of files 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 the deploy information.

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

Since Reakit bot is a project specific bot, and it is not accessible in the GitHub Marketplace, we could not use the Reakit bot in the experiment. Therefore, we replace the Reakit bot with two well-known and highly utilized bots:

- **Request-info** – requires more information on newly opened pull requests that contain the default title or blank description.
- **TODO bot** – creates comments on a PR based on the pending comments that exist in the code submitted in the pull request.

We created two scenarios for the experiment. Scenario 1 consists of two pull requests, A and B. Both are based on Reakit's pull request 796[4]. A developer created it in response to issue 745[5], a proposal to implement new features. All comments from humans and bots have been replicated in A and B with the distinction that B has its bots comments aggregated by our meta-bot. In turn, Scenario 2 is composed of pull requests C and D; both based on the pull request 828[6], a bug fix described in issue 827[7]. As in Scenario 1, all comments from humans and bots were replicated in C and D, distinguishing that C has the bot comments aggregated by the meta-bot.

Participants recruitment. Before recruiting the participants, we conducted sandbox experiments with one PhD student and four developers who are novices contributing to open source projects on GitHub to validate the script and confirm whether the experiment would fit in the time slot. For the experiment, 22 bachelor's and 3 master's students in Computer Science were recruited. Of the undergraduate students, there are 9 in Computer Science and 13 in Software Engineering. They were expected to have low or no experience with GitHub bots.

In total, we conducted 25 sessions with participants identified as P1 - P25. Table 5.1 presents the participants' demographics. Their experience contributing to OSS varies between none and experienced (with low experience on average). The experience with GitHub, on the other hand, varies from low to expert (with a reasonable experience on average). Finally, their experience with GitHub bots varies between none and experienced (with no experience on average). In addition, six participants reported having some experience with one or more GitHub bots, namely: dependabot, Vercel, analysis-bot and facebook-github-bot. All other participants (19) reported no previous contact with GitHub bots.

Experiment execution. We conducted the study via video calls since it enables us to guide participants throughout the experiment, which guarantees they follow the script order. Along with the invitation to the video call, the participants received by email instructions for the experiment and a survey with questions.

To mitigate bias, the distribution of scenarios and their respective pull requests occurred alternately. All participants started the experiment with Scenario 1, afterwards moved to

[4] https://github.com/reakit/reakit/pull/796

[5] https://github.com/reakit/reakit/issues/745

[6] https://github.com/reakit/reakit/pull/828

[7] https://github.com/reakit/reakit/issues/827

Participant ID	Course	Level	Experience level with*		
			OSS	GitHub	GitHub Bots
P1	Computer Science	Master	Low	Low	None
P2	Computer Science	Master	Low	Experienced	None
P3	Computer Science	Master	None	Reasonable	Low
P4	Computer Science	bachelor	Low	Experienced	Low
P5	Software Engineering	bachelor	None	Reasonable	None
P6	Software Engineering	bachelor	None	Experienced	None
P7	Software Engineering	bachelor	None	Low	None
P8	Computer Science	bachelor	None	Reasonable	None
P9	Software Engineering	bachelor	Low	Experienced	None
P10	Software Engineering	bachelor	None	Experienced	None
P11	Computer Science	bachelor	None	Reasonable	None
P12	Software Engineering	bachelor	Reasonable	Experienced	Expert
P13	Computer Science	bachelor	Low	Reasonable	None
P14	Software Engineering	bachelor	Low	Reasonable	None
P15	Software Engineering	bachelor	Experienced	Experienced	Low
P16	Software Engineering	bachelor	Low	Low	None
P17	Computer Science	bachelor	Experienced	Expert	None
P18	Software Engineering	bachelor	None	Experienced	Low
P19	Computer Science	bachelor	Experienced	Expert	Experienced
P20	Computer Science	bachelor	None	Reasonable	None
P21	Software Engineering	bachelor	Low	Reasonable	None
P22	Software Engineering	bachelor	Experienced	Experienced	Low
P23	Computer Science	bachelor	Low	Experienced	None
P24	Software Engineering	bachelor	Low	Reasonable	None
P25	Computer Science	bachelor	Low	Reasonable	None

* *None < Low < Reasonable < Experienced < Expert*

Table 5.1: *Demographics of participants*

Scenario 2. More specifically, thirteen participants started with pull request A from Scenario 1 (1-A), while twelve participants started with pull request B from the same scenario (1-B). When starting the session, we contextualized the participant about a scenario where they contributed to Reakit, submitted a pull request, and waited for some change in the status quo. This contribution concerns the implementation of a new functionality described in a current issue in the repository. The participant received a link to the pull request and was guided to use the information presented to identify its current status and the next steps to be taken for the pull request to be accepted by the maintainers. At the end of the analysis, we redirect the participant to a survey to capture their perceptions. After responding to the survey, we moved to Scenario 2. Once again, we guided the participant to look at the pull request to understand its current status. As for scenario 1, we redirected participants to a survey after they were done with the pull request analysis. This final survey aimed at comparing the two approaches: meta-bot and multiple bots. The group of participants who received 1-A and 2-C is referred to as Group 1 (G1), while 1-B and 2-D as Group 2 (G2).

Data Analysis. We used a card sorting approach (Zimmermann, 2016) to qualitatively analyze the answers to the open-ended questions. Two researchers conducted card sorting in two steps. In the first step, each researcher analyzed the answers (cards) independently and applied codes to each answer, sorting them into meaningful groups. This step was followed by a discussion meeting until reaching a consensus on the code names and categorization of each item. At the end of this process, the answers were sorted into high-level groups. In the second step, the researchers analyzed the categories, aiming to refine the classification and group-related codes into more significant, higher-level categories and themes. We used *open* card sorting, meaning we had no predefined codes or groups; the codes emerged and evolved during the analysis process. In addition, we quantitatively analyzed closed-ended questions.

5.1.2 Results

In this section, we focused on the results obtained comparing both approaches: using multiple bots *versus* using the meta-bot. Figures 5.3 and 5.4 report the main results.

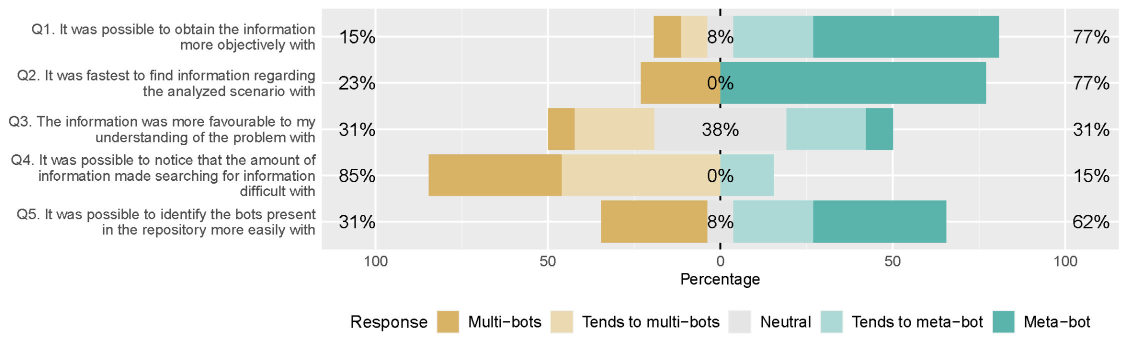


Figure 5.3: *Comparison results from Group 1*

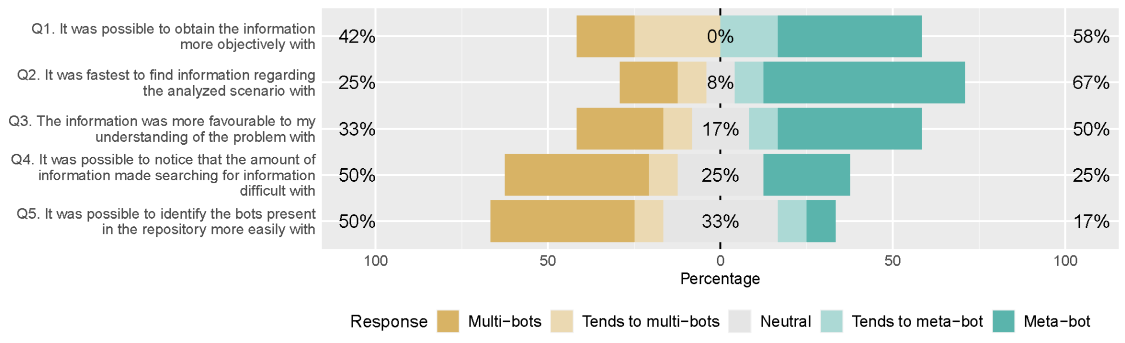


Figure 5.4: *Comparison results from Group 2*

Most participants in G1 and G2 (77% and 58%, respectively) pointed out the meta-bot approach as the option that allows obtaining information quicker. Also, most participants (77 % in G1 and 67 % in G2) found the meta-bot approach as the fastest way to find information regarding the analyzed scenario. In G1, a participant described a situation to justify his preference for the meta-bot: "*if I am responsible for deploying the application, I will know accurately and quickly where to look for information regarding my responsibilities in that repository [...]*". In G2, a participant reports that in the meta-bot, the "*information [is] aggregated in a comment containing all the relevant information generated by the bot*", which facilitates the access to such information.

Participants had different opinions related to which approach made it easier to understand the information. In G1, participants reported that the organization of the information has not affected their understanding. One participant mentioned that in situations where they know the category of the problem reported by the bot (e.g. deploy, critical), they prefer the meta-bot because of the comment categorization. However, if they need to analyze the entire pull request, they would prefer the various bots as they present the information along the pull request timeline. In G2, 50% of the participants found the meta-bot's approach as the easiest to understand the problem reported by the bots. One participant mentioned that the meta-bot "*has the same benefits as the various bots, however it aggregates the information, which helps to understand the information*".

In both G1 and G2, the participants (85% and 50%, respectively) said that the interaction with several bots made it challenging to search for information. Furthermore, in G1, most participants (62%) stated that when interacting with the meta-bot, it was easier to identify the bots present in the repository. A participant from this group reported that the way the meta-bot comment was structured drew attention and highlighted the bots commenting in the pull request. In contrast, in G2, a more significant number of participants (50%) stated that it was easier to identify the bots present in the approach with several bots.

Concerning the benefits of the meta-bot, the most frequent answer refers to its ability to **facilitate finding important information** (8 and 4 mentions, in G1 and G2 respectively). The participants justified this by mentioning the meta-bot **reduce the information overload** (3 mentions in G1), **categorize bot comments** (5 and 4 mentions, in G1 and G2 respectively) and **aggregate of comments** (5 mentions in both groups). One participant stated that "*for people used contribute to open-source projects daily, the [meta-bot] keeps the information aggregated into a single comment, giving more visibility to developers' comments.*"

> A preliminary evaluation of the meta-bot concept showed that aggregating bot comments was helpful to facilitate developers' to find the appropriate information from bots interacting on a pull request. However, as the meta-bot maintains the other bot comments as-is, it might not affect the understanding of the comment message when comparing to the multiple bots approach.

5.2 Participatory Design Fiction

In the following Sections, we present the research design and findings from our Participatory Design Fiction study.

5.2.1 Research Design

We devised a study[8] split into two phases, as depicted in Figure 5.5. 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

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

participatory design fiction methodology (Phase I). We describe the method and results from Phase II in Section 5.2.3.

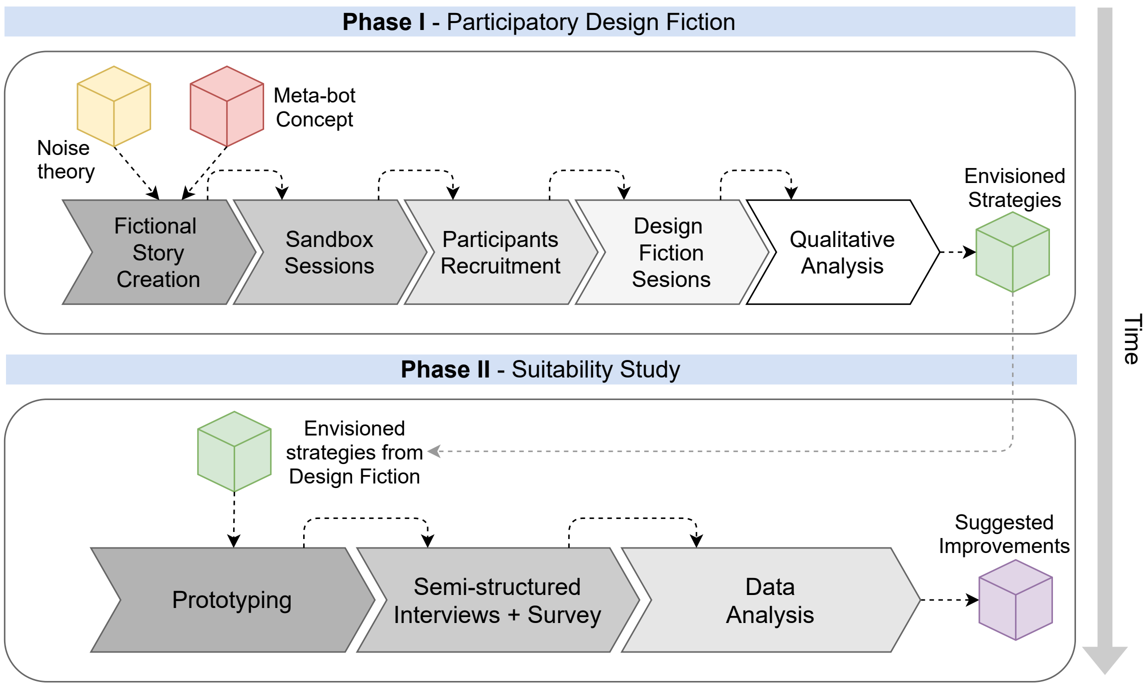


Figure 5.5: *Overview of the Research Design*

Phase I: Research Approach

We applied Design Fiction method (Blythe, 2014; Sterling, 2009), which has been broadly used in the Human-Computer Interaction field (Encinas and Blythe, 2016; M. Muller and Erickson, 2018; Blythe and Encinas, 2016). The Design Fiction method was first defined by Sterling (2009) as "*the deliberate use of diegetic prototypes to suspend disbelief about change.*" Design fiction 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 (Blythe, 2014; Lupton, 2017; Harmon *et al.*, 2017; Encinas and Blythe, 2016; Lindley *et al.*, 2016; M. Muller and Erickson, 2018; Linehan *et al.*, 2014). Researchers have been employed this method in an empirical way to elicit information from participants (Noortman *et al.*, 2019; Candello, Pichiliani, *et al.*, 2019) and communicate their insights (Fritsch *et al.*, 2013; Kirman *et al.*, 2013). The speculative nature of this technique amplifies critical views of current social and technological developments, creating a fictional context narrated through designed artifacts (Coulton *et al.*, 2017). This approach facilitates exploring boundless thoughts and open discussions on a particular subject (Blythe and Encinas, 2016). For instance, many researchers use design fiction to anticipate issues (Blythe and Encinas, 2016), while others focus on values related to new technologies (M. Muller and Liao, 2017; Cheon and Su, 2018) and anticipate users' needs (Cheon and Su, 2017; Encinas and Blythe, 2016; Noortman *et al.*, 2019).

Past studies applied Design Fiction as a participatory method to unveil design strategies for development technologies in a narrative format (Candello, Pichiliani, *et al.*, 2019;

Nägele *et al.*, 2018; Forlano and Mathew, 2014). According to M. J. Muller (2007), narratives in participatory work may be told by users as part of their contribution to specifying what products or services should do. Candello, Pichiliani, *et al.* (2019), for example, applied Design Fiction to explore the expectations of science museum guides when teaching robots how to answer museum visitors questions. Candello, Pichiliani, *et al.* (2019) crafted a fictional 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, Pichiliani, *et al.* (2019)'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 (Bleecker, 2004) to the envisioned technology (Kirby, 2010). 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.

Phase I: Method

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

The Fictional Story. The story description follows the key idea raised by the noise theory of Wessel, I. Wiese, *et al.* (2021): 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 the fictional story takes place approximately ten years in the future to let them be less constrained by current technological limitations.

After creating the fictional 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. Following, we present the *fictional story* that served as a baseline for the video creation. In addition, we made the video publicly available within the supplemental material.[9]

The Fiction: It is the year 2030. Ada is a maintainer of an open-source software hosted on GitHub called FutureX. Since FutureX receives thousands of monthly contributions, she spends considerable time maintaining issues and reviewing contributions. To reduce her workload with repetitive tasks and help her be more proactive, Ada relies on more than ten software bots to help her. One bot updates the project's dependencies to the latest version

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

when they become available, another bot makes contributors aware of code coverage, and another bot closes pull requests that have been inactive (stale) for a long time.

Ada recognizes that these bots reduce her burden with repetitive tasks and speed up the pull request review. However, she noticed that the other maintainers of FutureX complain about bot annoying behavior.

"*You are talking to the person who submitted the pull request and then a bot comes in and adds information in the middle of your conversation*" said Ellie.

John agreed. "*I agree with Ellie… Sometimes the bot comments are really verbose and overuse visual elements.*"

"*I feel the same,*" said Anne agreeing with both Ellie and John, "*these bots work a lot and generate so many notification for me to check.*"

After discussing with other project members, Ada concluded that the most recurrent problem is the introduction of noise into developers' communication channels.

"*The noise leads to an overload of information, which negatively impacts our communication and the development workflow*" complemented Ada.

Ada and her team started to think about a *Super Bot* to mitigate the noise caused by the existing bots. The *Super Bot* would orchestrate and mediate the action of other bots used on pull requests. You heard about this *Super Bot* and decided to help Ada's team to design it.

Sandbox Interviews We conducted sandbox sessions with a small sample of participants to adjust the fictional 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 would fit in a 1-hour time slot. The sandbox participants watched the video, answered all the questions, and provided us with feedback about the flow 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.

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 (Singer *et al.*, 2014; Storey, Treude, *et al.*, 2010; Aniche *et al.*, 2018), 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 design fiction sessions with 32 participants—identified here as P1–P32. Table 5.2 shows the demographic attributes of our participants. The majority (28)

Participant	Gender	OSS Experience	Location	Experienced with bots as			
ID		(years)		Bot dev.	Maintainer	Contributor	Researcher
P1	Man	5-10	Europe	✓	✓	✓	✓
P2	Man	4-5	South America		✓		
P3	Man	> 10	Europe		✓	✓	
P4	Woman	1	North America	✓	✓		✓
P5	Man	4-5	South America		✓	✓	
P6	Man	4-5	North America	✓	✓	✓	✓
P7	Man	5-10	Europe	✓	✓		
P8	Man	4-5	North America		✓	✓	✓
P9	Man	> 10	South America	✓	✓	✓	✓
P10	Man	3	North America	✓	✓	✓	✓
P11	Non-Binary	> 10	North America		✓	✓	
P12	Man	> 10	North America	✓	✓	✓	
P13	Man	5-10	North America	✓	✓	✓	
P14	Woman	3	Europe		✓		
P15	Man	4-5	South America		✓	✓	
P16	Man	> 10	North America	✓	✓	✓	
P17	Man	1	South America			✓	
P18	Man	4-5	Europe		✓	✓	✓
P19	Man	3	Europe		✓	✓	
P20	Man	2	South America		✓	✓	
P21	Woman	4-5	Europe		✓	✓	
P22	Man	5-10	North America	✓		✓	✓
P23	Man	5-10	South America	✓	✓	✓	
P24	Man	5-10	Europe		✓	✓	✓
P25	Man	> 10	North America	✓	✓	✓	
P26	Man	3	Europe		✓	✓	✓
P27	Man	5-10	Europe		✓	✓	✓
P28	Man	4-5	Europe		✓	✓	
P29	Man	5-10	Europe		✓		
P30	Woman	Zero	Europe				✓
P31	Man	1	Europe			✓	✓
P32	Man	3	North America	✓	✓		

Table 5.2: *Demographics of Design Fiction Participants*

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.

Design Fiction Sessions We conducted a series of synchronous design fiction sessions. The motivation behind this approach, instead of asking the participants to watch the story and write its end (M. Muller and Erickson, 2018; Candello, Pichiliani, *et al.*, 2019), was to engage the participants and ask questions during and after the debriefing. The sessions provided the flexibility to delve deeper into unforeseen information and enabled researchers to explore topics that emerged during the session (Hove and Anda, 2005). Before each session, we shared a consent form with the participants asking for their permission to video record. We also sent our participants a 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 based the scenarios in the noise theory presented by Wessel, I. Wiese, *et al.* (2021). Next, we present the investigated scenarios:

Scenario One (S1) – Newcomers. In this scenario, we describe a situation that occurs when a developer submits their first 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. We also created a scenario to describe 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 (Wessel, I. Wiese, *et al.*, 2021), 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, I. Wiese, *et al.* (2021), 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[10]. Each session was conducted remotely by the author of this dissertation and lasted on average 54 minutes. The participants received a 25-dollar gift card as a token of appreciation for their time.

Qualitative Analysis Each session recording was transcribed by the author of this dissertation. To qualitatively analyze the session transcripts, we applied open and axial coding procedures (A. L. Strauss and J. M. Corbin, 1998) throughout multiple rounds of analysis. We started by applying open coding, whereby we identified the envisioned features for the meta-bot or its environment. The author of this dissertation 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, the author and another experienced researcher discussed the emergent codes and reached an agreement in weekly hands-on meetings. During these meetings, the

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

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 (Patton, 2014). Then, the analysis was presented and discussed with the other three researchers. During the data analysis process, we employed a constant comparison method (Glaser and Anselm L Strauss, 2017), wherein we continuously compared the emerging codes from one session with those obtained from the previous ones. Afterward, the 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.

5.2.2 Phase I: Design Fiction Findings

In this section, we present the strategies to mitigate bots' noise derived from the analysis of the participatory design fiction sessions (see Section 5.2.1). The participants discussed several design strategies to mitigate noise created by bots, as presented in Table 5.3. 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 into five categories: *information management* (IMi), *newcomers' assistance* (NAi), *notification management* (NMi), *spam and failures management* (SMi), and *platform support* (PSi). In the following, we present these five main categories. We describe the categories in **bold**, and provide the number of participants we assigned to each category (in parentheses).

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,

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

Table 5.3: *Envisioned Design Strategies for the Meta-bot and the GitHub Platform (# = Number of participants we assigned to each category)*

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, five 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] fixes 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' first 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 your first 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 newcomer fixes 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.*"

> **Summary of phase I.** As a result of the design fiction 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.

5.2.3 Phase II: Suitability Study

To further refine the strategies elicited via the design fiction 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.

Phase II: Method

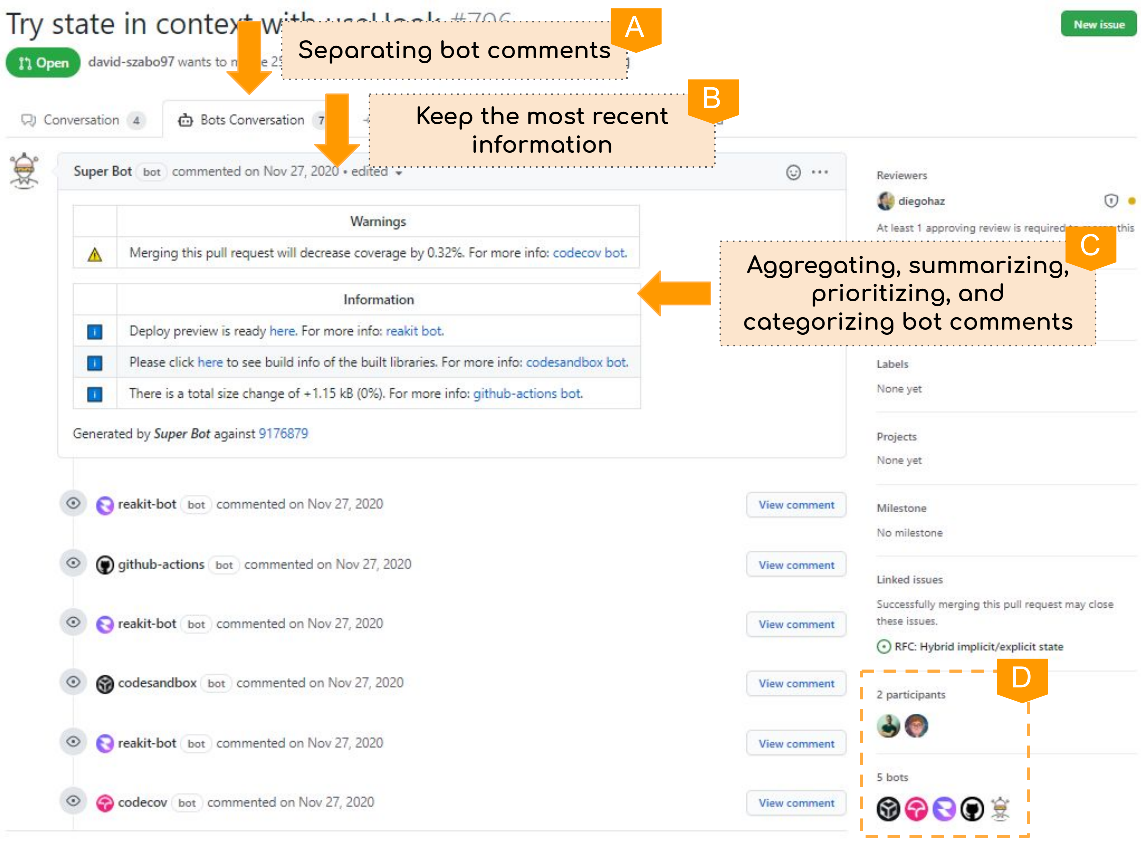
Prototyping We developed a prototype to receive feedback about the most cited strategies. We applied the prototype to a scenario created from the Reakit[11] project. This project uses four bots to support the pull request review process and all bots report their outputs using comments on the pull request. 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 of files 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.

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 5.6), designed to support maintainers and experienced contributors; and the newcomers' pull request interface (see Figure 5.7). In Figure 5.6, 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 5.6-A) called "Bots Conversation". This tab contains all information and events regarding bots in the pull request, including a timeline

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

of bot events. As for bot outputs, we also disambiguate the bot participants from human participants, as shown in Figure 5.6-D.


Separating bot comments
A
Keep the most recent information
B
Aggregating, summarizing, prioritizing, and categorizing bot comments
C
D
New issue
Open
Conversation 4
Bots Conversation 7

Try state in context with useHook #706

david-szabo97 wants to m… e 2…

Super Bot bot commented on Nov 27, 2020 • edited

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

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

Generated by *Super Bot* against 9176879

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

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

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

codesandbox bot commented on Nov 27, 2020 View comment

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

codecov bot commented on Nov 27, 2020 View comment

Reviewers
diegohaz
At least 1 approving review is required … this

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.
RFC: Hybrid implicit/explicit state

2 participants

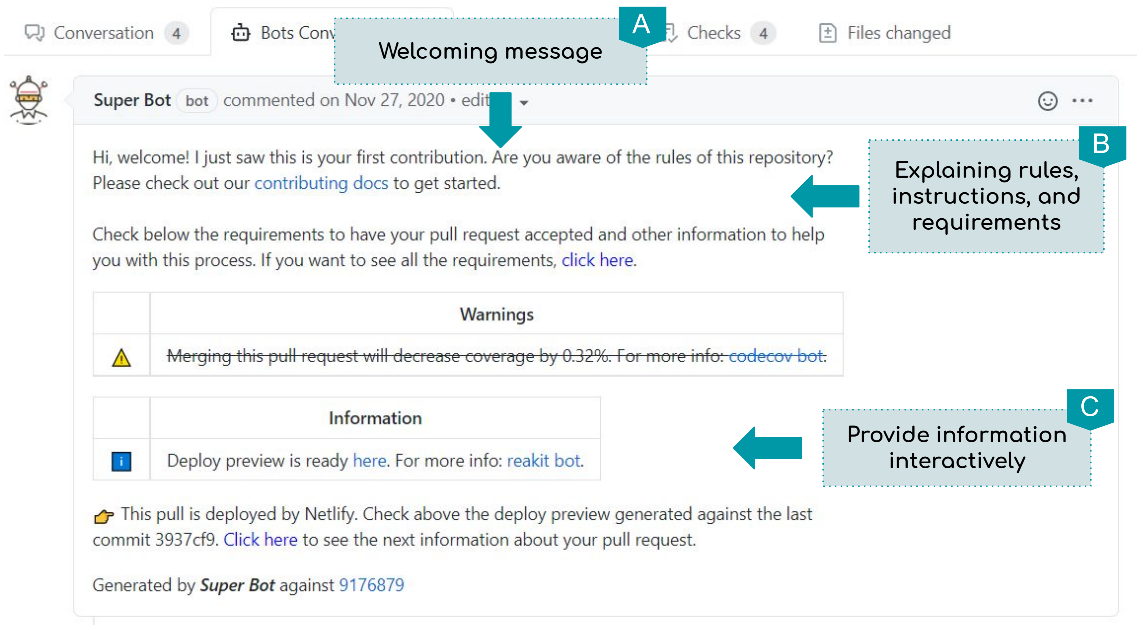
5 bots

Figure 5.6: *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 Figure5.6-C. 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 (Figure5.6-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 5.7 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 5.7-A. Beyond presenting a welcoming message, design fiction 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 5.7-B).

Another important distinction is the way the meta-bot displays the information for


Conversation 4
Bots Conv
Checks 4
Files changed
A
Welcoming message
B
Explaining rules, instructions, and requirements
C
Provide information interactively

Super Bot bot commented on Nov 27, 2020 • edit

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

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

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

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

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

Generated by ***Super Bot*** against 9176879

Figure 5.7: *Newcomers' pull request interface*

newcomers versus experts. In Figure 5.7-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.

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 our findings (Merriam, 1998), expressing their preferences about the elements of the designed prototype (Jeffery *et al.*, 2017). 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, the 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.

Technology Acceptance Model To assess the participants perception about the designed interface, we also applied the Technology Acceptance Model (TAM) (Davis, 1989) 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., (I. Steinmacher, T. U. Conte, *et al.*, 2016; Chen *et al.*, 2012)).

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 would find the designed interface useful.
Perceived ease of use - (PEOU)
E1. Learning to operate the designed interface would be easy for me. **E2.** I would find 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 would find 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.

Table 5.4: *Scale items for measuring usefulness, ease of use and self-predicted future use*

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 5.4 shows our assessment model which was adapted from previous literature (Babar *et al.*, 2007; Davis, 1989). 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.

Phase II: Results

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

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.*"

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

Table 5.5: *Suggested improvements to the prototype (# = Number of participants we assigned to each category).*

In addition to the positive comments, we found that some design elements needed improvements, as shown in Table 5.5. During the analysis we could identify seven potential points of improvement reported by the participants. Next, we further explain the reasoning behind those suggestions.

I*nclude 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 to filter 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.*"

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 (Bland and Altman, 1997). Carmines and Zeller (1979) 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.

Most participants found the designed interface useful. We present each item's detailed results in Figure 5.8. 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 (U1), easier job (U5), and usefulness (U6) had more than 85% of agreement or strong agreement.

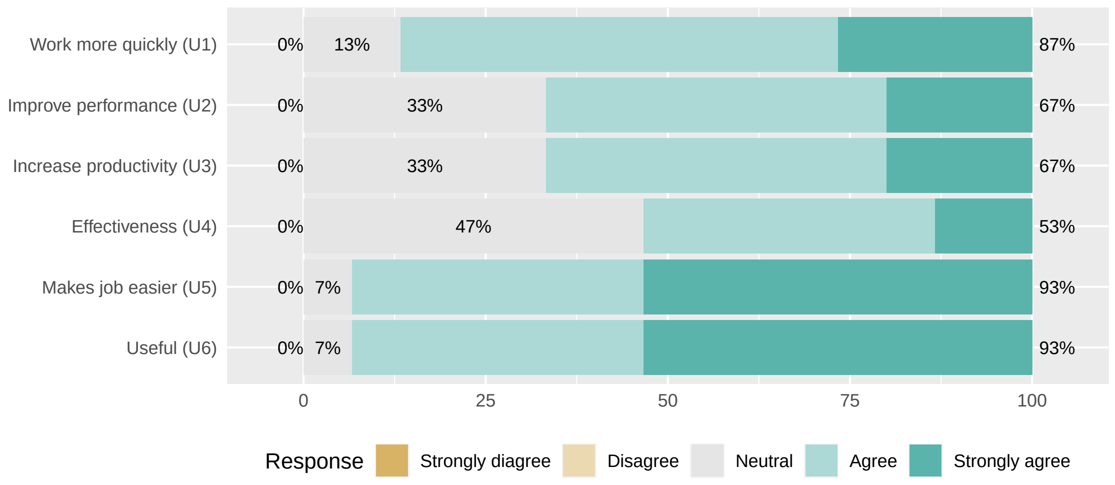


Figure 5.8: *Responses to the 5-point Likert-scale items for Perceived Usefulness*

In Figure 5.9, 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 5.2.3, we highlighted the suggestions to improve the design interface, which are likely to affect the ease of use positively.

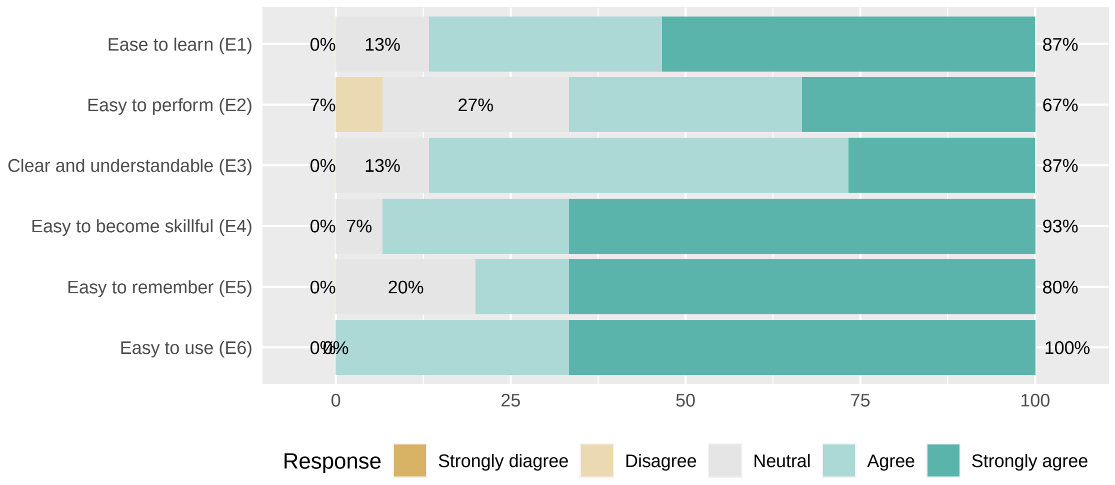


Figure 5.9: *Responses to the 5-point Likert-scale items for Perceived Ease of Use*

Figure 5.10 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.

Summary of phase II. We found seven potential improvements for our designed interface. Participants perceived the designed interface as a useful and easy to use

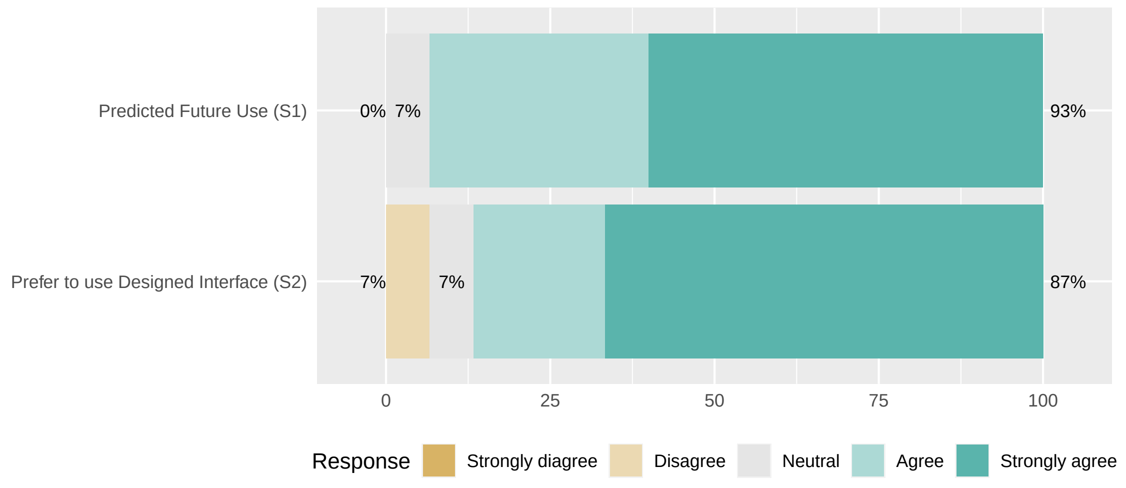


Figure 5.10: *Responses to the 5-point Likert-scale items for Self-predicted Future Use*

interface, and would potentially use it in the future, indicating the suitability of the design strategies.

5.3 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 (Wessel, I. Wiese, *et al.*, 2021; Erlenhov, Neto, *et al.*, 2016; Brown and Parnin, 2019; Mirhosseini and Parnin, 2017; Peng and Ma, 2019). Employing Design Fiction as a method to prototype a technology (Knutz and Markussen, 2014), 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 (Miller, 1956), concise and well-organized information might help developers to leverage bots outputs.

In line with Erlenhov, Neto, *et al.* (2016)'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 (Wessel, Souza, *et al.*, 2018). 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 education field learn from previous interactions and estimate students' interest level (Nakamura *et al.*, 2012) or learning styles (Latham *et al.*, 2010), 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, our findings 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 (Vorvoreanu *et al.*, 2019). 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 design fiction, 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, I. Wiese, *et al.* (2021) 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 more flexible 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.

5.4 Limitations and Threats to Validity

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

Generalizability of the results Since we recruited practitioners experienced with bots on the GitHub platform, our findings 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 design fiction 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.

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 (A. Strauss and Juliet M Corbin, 1997), 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 our findings, we employed a constant comparison method (Glaser and Anselm L Strauss, 2017). In this method, each interpretation is constantly compared with existing findings 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.

5.5 Final Considerations

In this Chapter, we took the first steps toward overcoming information overload created by bots. To do so, we implemented the meta-bot and conducted a preliminary study to validate its concept. A preliminary evaluation of the meta-bot concept showed that aggregating bot comments was helpful to facilitate developers' to find the appropriate information. We then decided go one step further, involving the users during the design process, to evolve the preliminary concept.

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 a fictional 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 the fictional story, raising concerns around the use of bots and discussing the design strategies to mitigate noise.

Grounded in participatory design fiction, 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, these findings provide a comprehensive understanding and exploration of design ideas to enhance the integration between bots, humans, and social coding platforms.

Chapter 6

Conclusion

The literature on bots on social coding platforms report several potential benefits, such as reducing maintainers' effort on repetitive tasks (Wessel, Serebrenik, I. Wiese, I. Steinmacher, and Marco Aurelio Gerosa, 2020) and increasing productivity (Erlenhov, Neto, *et al.*, 2016). In this dissertation, we extend previous work by delving into the challenges incurred by bots on social coding platforms. We identified, organized, and discussed the key effects and challenges of using and interacting with software bots on OSS projects' pull requests. We focused on the most recurrent challenge, namely noise, and proposed strategies to overcome this challenge. In this chapter, we summarize the conclusions of this work by bringing all of the results presented in previous chapters. Then, we discuss the implications for the design of future bots and the limitations of this research, leading to a set of directions for future work.

First, in Phase I of this dissertation, our goal was to understand the effects of adopting bots to OSS projects pull requests. We then investigated how several activity indicators change after the adoption of a bot. To do so, we focused on code review bots, which is one of the most common types of bots according with our warm up studies (Wessel, Souza, *et al.*, 2018). We employed a regression discontinuity design on 1,194 software projects from GitHub. Afterward, to further shed light on our results, we interviewed 12 practitioners, including open source maintainers and contributors. We found that, after code review bot adoption, more pull requests are merged into the codebase, and communication decreases between contributors and maintainers. Considering non-merged pull requests, after bot adoption projects have fewer monthly non-merged pull requests, and faster pull request rejections. From the practitioners' perspective, the bot comments make it easier to understand the state and quality of the contributions and increase maintainers' confidence in merging pull requests. According to them, contributors are likely to make changes in the code without interacting with other maintainers, which also helps to change the focus of developers' discussions.

In Phase II, 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. We relied on data collected from semi-structured interviews with 21 practitioners who have experience interacting with bots on pull requests. We validated our findings through member-checking. While participants commend bots for streamlining the pull request process, they complain about several challenges they introduce. In particular, our findings indicate noise as a recurrent and central problem. Frequently annoying bot behaviors such as verbosity, too many actions, and unrequested or undesirable tasks on pull requests, are perceived as noise.

In Phase III, we aimed at design strategies to mitigate the information overload created by annoying bot behaviors. Based on the hypothesis that *a meta-bot can mitigate the information overload created by other bots around pull requests*, we conducted an empirical study to assess a preliminary version of the meta-bot, which aggregate the outcomes of other bots on a pull request. Our findings indicate the meta-bot as a promising approach to mediate the existing bots. To further investigate the ideal features of the meta-bot, we used the Participatory Design Fiction approach to involve bot experts in the design process. Then, we identified several strategies that we used as insights to the implementation of a prototype. In the designed interface, we created a specific place in the pull request to display the bots' interactions. The meta-bot was evolved to provide a summarization of bot outputs, also prioritizing the most import ones. The results of a suitability study revealed seven desired improvements to the interface, such as creating bot references in the main conversation timeline to keep track of bot events that occurred during the pull request lifetime.

We also analyzed the perceived usefulness, ease of use, and predicted future use of the designed interface. These results are very encouraging, showing that participants perceived the prototype as a useful and ease-to-use tool to overcome noise. Participants also indicated a potential future use of the designed interface. We found very positive results with more than 53% agreement in all evaluated items. The item "*I would find the designed interface ease to use*" was positively evaluated by all participants.

6.1 Delivered Contributions

The artifacts generated in Phase I, II, and II of this dissertation represent the main contributions. In Phase I, we empirically identified the changes in project activity indicators after the adoption of a code review bot. We also elucidated the open-source developers' perspective on the impacts of code reviews bots. These contributions aim to help practitioners and maintainers understand, or even predict, bots' effects on a project, especially to avoid the ones that they consider undesirable. Additionally, our findings may guide developers to consider the implications of new bots as they design them.

During Phase II, we empirically identified the the challenges introduced by the interaction of bots on GitHub pull requests. We created a hierarchical categorization that summarizes these challenges. 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 scarce and diffuse challenges, reported as secondary results. Since noise emerged as a central theme in our analysis, we further theorize about it, grounded in the data we collected. Thus, the main contribution of Phase II is a theory of how human developers perceive annoying bot behaviors as noise on social coding platforms. This theory 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. More specifically, we provide the first step towards understanding how the information overload bots' cause might affect how humans communicate, work, and collaborate on social coding platforms.

In Phase III, we built a prototype of the designed interface, which comprises the meta-bot and several modifications to the GitHub's pull request user interface to fit the envisioned features. The meta-bot and the modifications to the user interface were based on the participatory design fiction study conducted with practitioners. We then conducted a suitability study of the designed interface with a subset of our interviewees and the results are very encouraging. Researchers and tool designers may also leverage our results to enhance bots' communication design, thereby better supporting human-bot interaction on social coding platforms.

We also claim that this dissertation contains some minor or secondary contributions, as follows:

Characterization of the usage of bots in OSS projects presented in the introduction and used to defined the scope of the dissertation is novel. In order to delimit the scope of this dissertation, we conducted an exploratory study to classify the bots supporting a set of 351 open-source projects. From our data analysis, we make the following contributions: (i) bring attention to bots, a relevant resource that offers support for collaborative tasks in open-source development; (ii) characterize the usage of bots in open-source projects; and (iii) elucidate how contributors and maintainers see the importance and support of bots.

The method used to design the Meta-bot and its environment The participatory design fiction (PDFi) approach is new to software engineering studies and can be reused in other works to support the design process of new technologies. We did not evaluate or discuss the method at length in this dissertation, but as it is a well established method in the human-computer interaction (HCI) field, we consider it a secondary contribution to the literature of software engineering.

6.2 Future work

The work presented here is the initial step toward designing approaches to overcome the information overload incurred by bots' interactions. In the following, we point to some important open challenges that can be addressed in future work:

Another look at the bot interaction challenges. Both noise theory and the reported general challenges are grounded in the qualitative analysis of data from practitioners who are experienced with bots on the GitHub platform. Hence, the bot interaction challenges can be further investigated and evaluated from the perspective of other social coding platforms, such as GitLab and Bitbucket.

Mining software repositories to confirm the information overload. Some of the identified annoying bot behaviors might be further analyzed using software mining techniques. A possible future direction could be to use these techniques to verify which annoying bot behavior a given project presents. This kind of study can be used as one of the indicators of information overload in a project.

Migration to other automation tools. Recently, GitHub introduced GitHub Actions, 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. In the GitHub Actions study we performed, we noticed that a considerable amount of projects have been switching from bots and other automation tools to Actions. Researchers could analyze how this migration affect and influence the perceived information overload, since the way these Actions communicate in the GitHub platform is the same as bots.

Different perceptions of noise. In the noise theory, we discovered that although annoying bot behaviors are the source of noise, the perception of such noise varies. For instance, noise perception depends on the experience and preferences of the developer interacting with the bot. Thus, a clear future direction is to explore in more depth how different developers, or even communities, perceive noise generated by bots and copy with its effects.

User Experience Assessment of the designed interface. As another future work, we plan to apply more robust user experience assessment methods to evaluate how open-source developers perceive and use the designed interface. We aim at evaluate whether the designed strategies in fact mitigate noise. Also, we wanted to receive feedback from developers belonging to various roles in the open-source development process: project maintainers; experienced contributors; and newcomers.

In conclusion, we expect that the results presented in this dissertation open new research avenues that bring together researchers in software engineering (ES), computer supported cooperative work (CSCW), and human-computer interaction (HCI) field to design bots that provide an enriched user experience, thereby mitigating the occurrence of information overload on the human communication channels.

Appendix A

Supplementary Material for Chapter 3

In this appendix, we present the complementary material to replicate the method available in the paper "*Quality Gatekeepers: Investigating the Effects of Code Review Bots on Pull Request Activities.*" For more details, refer to our Zenodo repository.[1]

A.1 Interview Guide

Demographic questions

- What gender do you identify as?
- Which continent are you currently living in?
- Do you maintain an open source project that uses code review bots?
- How many years of experience do you have as a maintainer of this project?

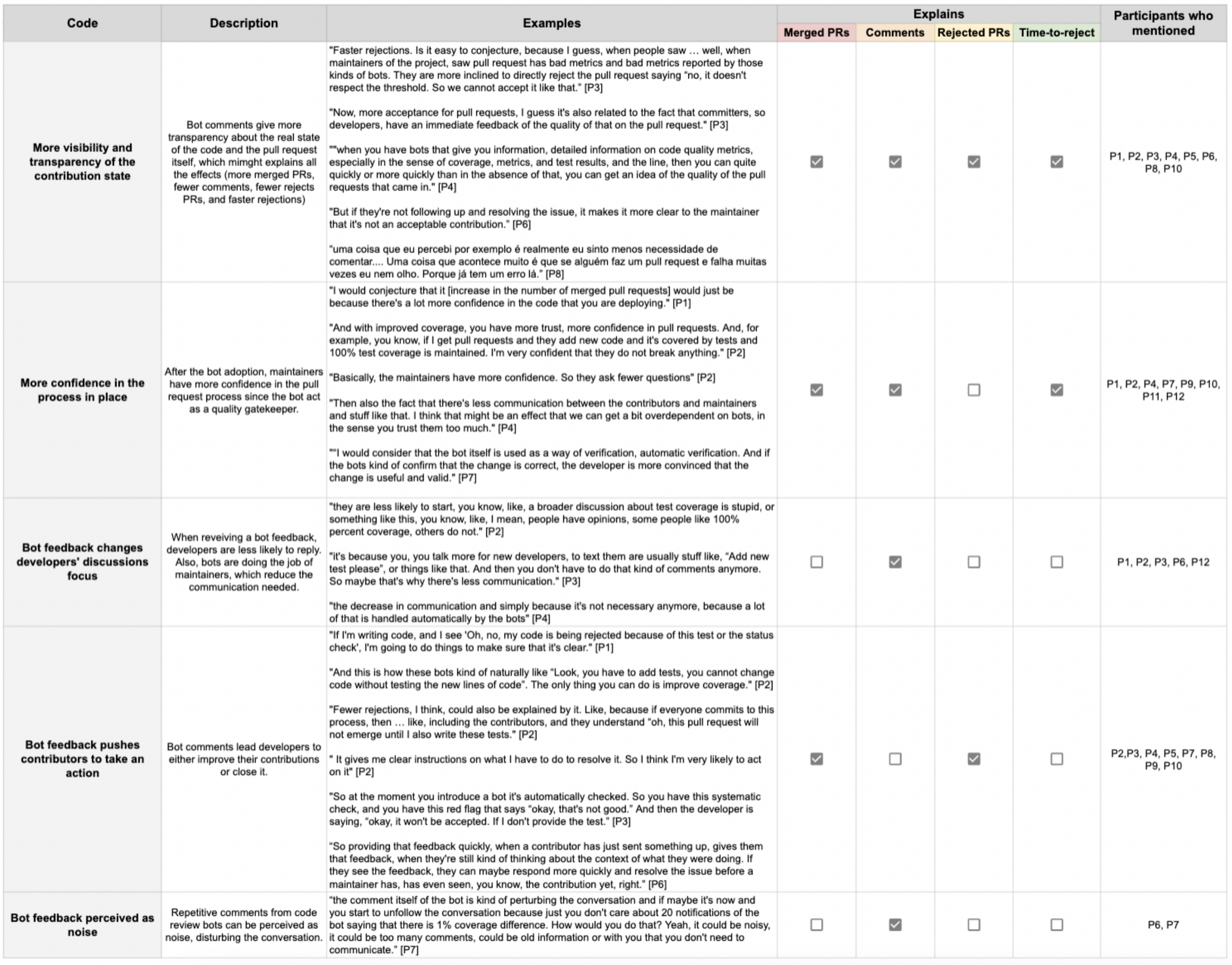
Explaining the statistical results We conducted a scientific investigation by mining data from open source repositories. Interestingly, we found that after adopting a code review bot there are more merged pull requests, less communication between developers, fewer rejected pull requests, and faster rejections. We are intrigued about these effects and would like to hear thoughts from developers who actually use these bots.

Follow-up Questions

- Could you conjecture the reasons why this happens?
- Have you observed these effects in your own project?
- What other effects did you observe in your project and attribute to the introduction of the code review bot?

[1] https://zenodo.org/record/4618499

A.2 Code Book



Code	Description	Examples	Explains				Participants who mentioned
			Merged PRs	Comments	Rejected PRs	Time-to-reject	
More visibility and transparency of the contribution state	Bot comments give more transparency about the real state of the code and the pull request itself, which mimght explains all the effects (more merged PRs, fewer comments, fewer rejects PRs, and faster rejections)	"Faster rejections. Is it easy to conjecture, because I guess, when people saw … well, when maintainers of the project, saw pull request has bad metrics and bad metrics reported by those kinds of bots. They are more inclined to directly reject the pull request saying "no, it doesn't respect the threshold. So we cannot accept it like that." [P3] "Now, more acceptance for pull requests, I guess it's also related to the fact that committers, so developers, have an immediate feedback of the quality of that on the pull request." [P3] ""when you have bots that give you information, detailed information on code quality metrics, especially in the sense of coverage, metrics, and test results, and the line, then you can quite quickly or more quickly than in the absence of that, you can get an idea of the quality of the pull requests that came in." [P4] "But if they're not following up and resolving the issue, it makes it more clear to the maintainer that it's not an acceptable contribution." [P6] "uma coisa que eu percebi por exemplo é realmente eu sinto menos necessidade de comentar.... Uma coisa que acontece muito é que se alguém faz um pull request e falha muitas vezes eu nem olho. Porque já tem um erro lá." [P8]	✓	✓	✓	✓	P1, P2, P3, P4, P5, P6, P8, P10
More confidence in the process in place	After the bot adoption, maintainers have more confidence in the pull request process since the bot act as a quality gatekeeper.	"I would conjecture that it [increase in the number of merged pull requests] would just be because there's a lot more confidence in the code that you are deploying." [P1] "And with improved coverage, you have more trust, more confidence in pull requests. And, for example, you know, if I get pull requests and they add new code and it's covered by tests and 100% test coverage is maintained. I'm very confident that they do not break anything." [P2] "Basically, the maintainers have more confidence. So they ask fewer questions" [P2] "Then also the fact that there's less communication between the contributors and maintainers and stuff like that. I think that might be an effect that we can get a bit overdependent on bots, in the sense you trust them too much." [P4] ""I would consider that the bot itself is used as a way of verification, automatic verification. And if the bots kind of confirm that the change is correct, the developer is more convinced that the change is useful and valid." [P7]	✓	✓	☐	✓	P1, P2, P4, P7, P9, P10, P11, P12
Bot feedback changes developers' discussions focus	When reveiving a bot feedback, developers are less likely to reply. Also, bots are doing the job of maintainers, which reduce the communication needed.	"they are less likely to start, you know, like, a broader discussion about test coverage is stupid, or something like this, you know, like, I mean, people have opinions, some people like 100% percent coverage, others do not." [P2] "it's because you, you talk more for new developers, to text them are usually stuff like, "Add new test please", or things like that. And then you don't have to do that kind of comments anymore. So maybe that's why there's less communication." [P3] "the decrease in communication and simply because it's not necessary anymore, because a lot of that is handled automatically by the bots" [P4]	☐	✓	☐	☐	P1, P2, P3, P6, P12
Bot feedback pushes contributors to take an action	Bot comments lead developers to either improve their contributions or close it.	"If I'm writing code, and I see 'Oh, no, my code is being rejected because of this test or the status check', I'm going to do things to make sure that it's clear." [P1] "And this is how these bots kind of naturally like "Look, you have to add tests, you cannot change code without testing the new lines of code". The only thing you can do is improve coverage." [P2] "Fewer rejections, I think, could also be explained by it. Like, because if everyone commits to this process, then … like, including the contributors, and they understand "oh, this pull request will not emerge until I also write these tests." [P2] " It gives me clear instructions on what I have to do to resolve it. So I think I'm very likely to act on it" [P2] "So at the moment you introduce a bot it's automatically checked. So you have this systematic check, and you have this red flag that says "okay, that's not good." And then the developer is saying, "okay, it won't be accepted. If I don't provide the test." [P3] "So providing that feedback quickly, when a contributor has just sent something up, gives them that feedback, when they're still kind of thinking about the context of what they were doing. If they see the feedback, they can maybe respond more quickly and resolve the issue before a maintainer has, has even seen, you know, the contribution yet, right." [P6]	✓	☐	✓	☐	P2,P3, P4, P5, P7, P8, P9, P10
Bot feedback perceived as noise	Repetitive comments from code review bots can be perceived as noise, disturbing the conversation.	"the comment itself of the bot is kind of perturbing the conversation and if maybe it's now and you start to unfollow the conversation because just you don't care about 20 notifications of the bot saying that there is 1% coverage difference. How would you do that? Yeah, it could be noisy, it could be too many comments, could be old information or with you that you don't need to communicate." [P7]	☐	✓	☐	☐	P6, P7

Appendix B

Supplementary Material for Chapter 4

In this appendix, we present the complementary material to replicate the method available in the paper "*Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects.*" For more details, refer to our Zenodo repository.[1]

B.1 Interview Guide

Starting the interview

- Remembering the terms of the consent form.
- Explaining research objectives.
- Asking permission for recording.

Questions about the participant's experience

- How experienced are you with software development?
- How experienced are you with open-source software development?
- What is your experience with GitHub bots (as maintainer, contributor, or bot developer)?
- For how long have you been interacting with software bots on GitHub?
- (OR - for bot developers) For how long did you maintain this bot?
- Do you have experience using bots in projects outside GitHub? (Yes/No)
  - Which type of bot?
  - How many years of experience do you have?

[1] https://zenodo.org/record/4443841

Questions about software bots on pull requests

- (For bot developers) How would you describe the trajectory of the design of these bots?
    - Which type of challenges have you experienced?
- Do you have experience in projects that use bots to support pull requests? (Yes/No)
    - How many projects? Which projects?
    - How many bots? Which bots?
- Have these bots been developed specifically for those projects? (Yes/No)
    - (For bot developers) Why did you build this bot?
    - (If Yes) Are they available for other projects/communities?
- How would you describe the role of these bots (the tasks they have to fulfill, purpose)?
- How do these bots interact on pull requests (commenting, labeling, assigning reviewers, opening/merging PRs)?
- Is it common to have more than one bot interacting in a pull request? (Yes/No)
    - (If yes) Can you tell me an example of multiple bots interacting?
- What kind of support do you expect from a bot adopted to work on pull requests?
    - Does it depend on the role of the bot?
- Do you think that the bot comments on pull requests are meaningful in providing feedback? (Yes/No)
    - (In both cases) Why?

Questions about human-bot interaction problems

- What is your perception about the ecosystem of bots working on GitHub platform?
- In your opinion, what are the main problems that the use of bots on pull requests introduces?
- What is the frequency of these problems?
- What is the relevance of these problems?
- Who do these problems mainly impact (contributors, maintainers, newcomers, ...)?
- Are these problems caused by a specific type of bot? (Yes/No)
    - (If Yes) Which one?
    - (In both cases) Why?
- Do bots introduce any technical issues to the project (configuration, . . .)?
- Do bots introduce any social or ethical issues to the project (communication, . . .)?

- Have you ever heard developers complaining about bots (about the interaction of these bots)?
- Do you remember any case of a bot bothering you? (Yes/No)
  - (If Yes) How did you handle it?
  - (If Yes) Did you or someone else solve this problem?
- Do you think these problems can lead projects to stop using a bot? (Yes/No)
  - (In both cases) Why?
  - (If Yes) Do you remember any?

Questions about solutions

- How would you change the bots to avoid such problems?
- How do you see the future of software bots for GitHub pull-requests?

B.2 Code Book – Noise Theory

Code			Description	Participants who mentioned	Examples
Factors	**Bot failure**	**Recovering from a failure**	When recovering from a failure the bot might leave all missed comments.	P3	it comment on issues when it was released and when it was back it create a thousand of comments. So, that is annoying if that happens. That is a big problem. But it is obviously a bug. I mean, it is annoying but it is not intended. [P3]
		Bug	When a bug occurs the bot might inflate the project with pull requests or comments	P6, P7, P19	So like, something went wrong with the release process. So it opened up a bunch of different pull requests. And like some of them were a mistake. So my other engineer that had to comment and be like, “Hey, sorry, like, these were a mistake.” [P6] The only times I've perceived our bots as noisy is when there's an obvious bug in the state transitions. [P7] (from member checking)
	Unforeseen problem with bot adoption		During the installation, the bot can apply the rules to all pull requests or all projects becoming noisy.	P3, P10, P21	Or maybe, when you install it for the first time and it will comment on every issue that is open and no longer… but, this is what you want, but it is also a lot of noise for everyone who is watching the repository and they might be upset about it and say “oh, this bot is stupid I will no watch this repository anymore.” [P3]
	Bot development	**Testing in a real scenario**	Developers often need to test bots on a real project, which might cause unexpected bot actions	P2, P12	It [the bot] was in development, but we put it in several repositories to test it. And it caused a very annoying problem. [...] And it broke the process a little bit, it was kind of boring. [P2] We struggle with, like, environments, I guess I would call, which I think is a little bit related to this testing in a real scenario. It's very... especially if your bot is designed to interact with people and other actions in the repository. It's just actually quite hard to do that testing. [P12] (*from member checking*)
		Bot deployment	During bot deploument, the bot might perform unexpected actions	P7	And then when we were trying to upgrade the bot, there were two versions running and they got an edit war sending me like 45 emails or something like that. Because they kept moving then ... so bot instance one was moving here and then bot instance two moved it back and advances to this. [P7]
	Bot design	**Spam**	When opt-out bot interact on a repository	P4, P16	I didn't give [to the bot] explicit authorization to enter the repository, it generates a notification that you did not request, which is the pull request. [P16] [...] people want to have agency, they want to have choice. [...] They want to know that they are being corrected because they asked to be corrected. So when I asked for you to do code coverage, and you tell me I'm failing my code coverage, I asked for that.[P4]
		Intended behavior	Some default bot behaviors can be perceived as noise	P4, P7, P19	But by default, it also leaves a comment. [P19]
Bot annoying behavior	**Verbosity**		Bots providing dense information and overusing graphical elements	P1, P13, P17, P19, P21	So the default is it puts a gigantic comment with a treemap on your pull request. People hate that. [P13] And so it has a GitHub integration 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. [P17] These bots that are chatty. [P21]
	Frequency and timing of actions	**Repetitive actions**	When a bot create a new pull request even if there are already an open pull request not addessed.	P3, P6, P9, P10, P14, P15, P19, P21	sometimes like maybe it's an automation that like runs too frequently or something and then it's like, “oh, it keeps opening up all the pull requests that I don't need or want me to close them.[P6] Sometimes we get comments like "hey, bot comments much to my taste" [P9] I guess if dependabot every week opening a bunch of pull requests was annoying before we automated it to then have a bot go close the pull requests. We solve that problem with another bot. [P15] It's very repetitive and it's possibly information dense. [P19]
		Time insensitive	Some bots actios are time insensitive, since they work all day long.	P3, P19	As long as, for example, the comment is immediately after I did a change, for example, when I merge the pull request and probot have the welcome bot and says “hey the pull request was merged”, and as long as it happens, if it is in a second or two and I’m seeing the page I do not get a new notification, because I already saw it and I’m still on the page. But if it happens on 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. [P3]
	Performing unexpected actions		Bots might perform unsolicited taks (e.g., spams)	P4, P6, P13, P16	So like, something went wrong with the release process. So it opened up a bunch of different pull requests. And like some of them were a mistake. So my other engineer that had to comment and be like, “Hey, sorry, like, these were a mistake.” [P6]
Bots overopulation			Too many bots working on the same repository	P7, P8, P9, P12, P13, P15, P19, P21	if you have a project with a lot of bots on.There can be a lot of notifications and interactions from that bots. [P12] ... a bunch of like stupid agents commenting on the pull request ... [P13] Obviously, we might get to the point where there's too many [bots]. And like that will be a problem. [P8] Yeah, because there were 30 different bots. And like each one of them was asynchronously going in. So it was just like, giving us tons and tons of comments. [P19]
Human previous experience			Different perceptions based on the previous experience of the developer (e.g., newcomer, experienced maintainer)	P3, P7, P14, P19	But if you, you know… when it is your self maintained project, and you see these comments everywhere and you can not configure the codecov to turn it off, it might become just noise. [P3] There's other stuff where it's really more for novices or something like [bot] tends to have pretty, pretty dense messages. [P19] Bots are too noisy for me. [P3] since I'm experiencing in bots, I think it's fine, but it can... Some can probably generate a lot of noise as well. [P14] I do worry that newcomers perceive the bots as noisy, even with only 1-2 comments, because the comments are large. [P7] (*from member checking*)
Project standards			When the bot is not compliant to the project standards	P8, P9, P14	So if the code is public, then the biggest fear of the public maintaining… maintainers of the bug report is like the bot is not compliant to their standards, right? 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. [P9] Like, my project isn't that complicated. I know which files I changed. And so if I'm not using it, then there's no point in seeing it. [P8] So, like dependabot and renovate that generate the updates, not all software projects follow the same version and convention sometimes it is easy to get it slightly wrong. It depends... you could set it just to update stable releases, but then you're a bit behind. So you sort of has to be... it's easy to be on like the bleeding edge with bots, I would say. [P14]
Information overload	**Notification overload**		Bots creating too many notifications (e.g., emails, GitHub notification)	P3, P12, P13, P16	So, suddenly I would be a 1000 of new notifications just because a bot and it would be very annoying to me because I would know a legitime new notification, something I need to see or a stupid comment from a bot. [P3] Nobody wants to be notified like 16 times about the pull request, and they'll just unsubscribe. [P13] [...] it creates basically new notifications about things that are not new to me to see [P3] who's not very active on GitHub will get completely overwhelmed by the review process because of all the notifications and they just… [P12]
	Replicate information for maintainers		Bots sometimes overload maintainers with information they already have (e.g., redundant information)	P3, P8, P10, P13, P17	I remember that codecov was integration and always left comments telling you “oh, the code coverage drop for 1%” or whatever. And it would become noise at times, like… I would prefer not to have the comments, for me as a maintainer I would… Because it replicates information that we already had. [P3] sometimes it's just commenting something that's redundant. [P17] some people don't care to get that extra information. [P8]
Disruption on the human communication	**Miss important information from humans**		Information overload can lead developers to miss important information from humans	P1, P10	Miss important comments from humans [P1] Before the bots, I could see who was working on the project in an easier way. For large projects it can be confusing, because do I really have several problems or are they just observations of the bots? [P10]
	Difficult to handle the information		It is difficult to handle all the information generated by bots	P8, P10	As a human, it's difficult to parse through all that data and get something meaningful out of it. [P8] A bot creating a lot of information, following its own patterns [...] To review all this content also takes a while. [P10]
	Interrupt the coversation flow		Since bots are using the same communication channel as developers, they interrupt the conversation flow	P13, P19	Because it interrupts. It's like you're 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] if the bots make just tons of comments and interrupt the conversation on the pull requests, we don't like that. [P13] I think the main problem was that I've ever experienced with bots has been spamming us and drowning out conversations. [P19]
	Burden to filter out notifications		Due to the amount of notifications, maintainers need to filter out human notifications from bot ones	P3, P16, P19	I don't want a lot of bot generating noise. I have to filter what is noise and what are important things and what are emergency things. It's like spam from GitHub notifications. [P16] I get an email like tries to grab my attention. Whereas the CI status hook, I will go back, you know, I like to differentiate between a real person is talking to me and here's status information that you need. [P19]

Disruption on the development workflow	**Time spent with non-development tasks**	Increase the burden for developers. Also waste developers time.	P5, P16, P20	it [the bot] spends extremely unnecessary time, from a person who is the maintainer of an Open Source package, who often does this voluntarily ... With noise, you know? For me it is a disaster. [P16] Before closing, it comments to ask if the problem is resolved. I then have to respond that it is not resolved. It's bothersome because there's no real progress being made, yet it still takes my time. [P20] It's an annoyance because it wastes our time. Constantly. We have to close it … we have to see "Oh, this is a PR", "Oh, this is a spam"... like I waste five minutes determining, determining that it is a spam. [P5]
Noise		Participants mentioned bots introducing noise	P1, P3, P7, P8, P9, P12, P13, P14, P16, P19, P20, P21	Noise. [P1] Some of them can introduce a lot of noise. [P3] So at some point, they [the bots] might be just too noisy. [P9]
Perceived as		[relationship] Some bot behaviors might be perceived as noise	P1, P7, P12	Depends on the verbosity of the bot and what it does. [P1] I do worry that newcomers perceive the bots as noisy, even with only 1-2 comments, because the comments are large. [P7] (from member checking) 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. [P12] (from member checking)
Countermeasures	**Stop watching a repository**	Contributors stop watching a repository to manage the noise	P8,P13, P19	And I certainly would stop watching. Like, I feel like I would stop watching repositories that got five or 10 dependency upgrades, pull requests every single day, or, you know, unless I had some, I would have to use some sort of aggressive filtering. It might destroy my, like ability to use GitHub notifications, things like that. [P19]
	Re-configuring the bot	Maintainers need to re-configure the bot to avoid noise	P8, P17, P19, P20	A significant nonzero number of them actually turn them [bot comments] off. And they turn them off because for them, it is not the right workflow for them. And so they prefer to, for us to only update status checks. [P8] So typically I'll just turn off those comments entirely. [P17] Re-configuring the bot to lower the frequency of actions. [P20] *(from member checking)* Sometimes we need to re-configure the bot. [P19] (*from member checking*)
	Re-designing the bot	Bot developers need to re-desing the bot to avoid noise	P12	So we did effectively redesign the messaging. [P12] *(from member checking)*

B.3 Code Book – Hierarchical Categorization

Code			Description	Participants who mentioned	Examples
Bot development issues	Platform limitations	Restricted bot actions	There are limitations on the API that limit the bot actions	P5, P10	"There are limitations to the API. Most of the things I want to do is possible to be done. But there are still a few things that just cannot be done with the API. So that's a problem that I face" [P5] if they make a PR I can directly adjust that PR myself like the code… That ability is not available through the API. I cannot build the bot. How do I say this? I cannot emulate that using API calls, it's GitHub limited the bot so that the bot cannot push a commit to the PR. [P5]
		Restricted bot communication	There are limitations on the API that limit the bot communication	P3, P4	ideally I think that the platforms would offer more integration points for these bots, because basically to communicate bots have use comments. [P3] Part of the problem is that the bot have to interact with the tools it has. So, it has comments and it has pull requests. And that is really the only sort of interaction it can do. [P4]
	Technical overhead to host and deploy bots	-	Maintainers face technical overhead to host and deploy their own bot	P7, P13, P14, P19	there's been technical overhead to maintain the bot. For a while the bots was running on a phisical machine. [P7] I supposed maintaining them could be a problem. Like, I mean, when someone has to spend time maintaining the bot itself, at some point there is going to be a question. There is some trade off, right? The bot saves you time but it also costs time to maintain. [P19]
	Challenges in building complex bots	-	Maintainers face challenges to build bots that are complex (e.g., using natural language)	P4, P11, P12, P18	I think a lot the ways people are building bots, they just automating a single thing. We don't have that. We just have one bot that does everything. I think it is hard to build a bot that has lot of capabilities. [P12] As a developer, I tried to build some bots using natural language processing. It is challenging. [P18]
Bot adoption issues	Discoverability issues	Difficult to find an appropriate bot	Project maintainers face some challenges to find an appropriate bot for their projects	P4, P6, P9	It's a bit harder to find the right application. And also the discovery of the application is how to say not so transparent because I don't know how users find our applications. [P9] From one it is discovery, right? There are actually really good ideias people could do with a bot and GitHub. But, as a developer, I don't know and I do not have time to search for it. [P4] there is a lack of awareness that you like can use bots and that like bots are available to you... to like help you. So, I think like a lot of people are like… are doing things they like don't need to be doing because like, they could have a bot do that for them, but they don't realize that's an option. I think that's one thing. [P6]
		Limited search mechanism for bots	There are limited mechanisms of search for bots on GitHub	P4, P6, P10	Primarily I do and part of the reason is, there really isn't a good discovery mechanism for other bots. [P4] I don't think it's very clear at all, like you can even use a bot or like how to use a bot or like where to find bots. In the marketplace, I don't know actually how much people use that. I don't even know if there is a category for bots. [P6]
	Managing bots' configuration	Difficult to get a tailored configuration	Project maintainers face some difficults to ajust the configurations for bots	P1, P8, P10, P16	We do see users who, you know, struggle to get the right configuration for them. [P8] It is easy to install the bot with the basic configuration. However, it is not easy adjust the configuration to your needs. [P10] Managing the configuration for bots [P1]
		Limited configuration	There are some limitations in the configurations for bots (e.g., how to configure more then one project)	P9, P10, P14	it could be hard to add to all the projects if you have a large number of projects, for example. [P14] So some of them just allow you to add the whole organization. Others enable you to like a few projects at a time for example. I think it also for example, I mean bot could analyze your git code organization and add new projects. Some doesn't. [P14]
		Burden to set up configuration files	Project maintainers need to learn how to configure the bot using a configuration file	P4, P15	It is like a whole configuration file you have to write. That is a lot of work, right? [...] Anytime I have to learn a new configuration file, I don't like it. It sucks. [P4] Now, instead of having user interface that I can go click on I have to read a bunch of docs and figure out how to make a config. [P15]
	Convincing developers to use the bot	-	Project maintainers/Bot developers face challenges convincing developers to use the bot	P4, P10, P14	We have some really good tools we could leverage for developers, but they're not gonna want to just use these tools randomly. And how do you convince them to use it? I don't know. I think that's the biggest problem for bots in general. [P4] I've induced them in a large scale at a place I worked, but I think like it… Initially you have to have on sort of opt in strategy to get people to adopt it. And if people like it, I think it will spread to the company. [14]
	Technical complexity issues	Bot increase the barrier for new maintainers	Bot might increase the complexity for new project maintainers	P3	I think they can introduce complexity to the project and increase the barrier to onboarding new maintainers. If these bots need to be configured and you need to understand how they work [P3]
		More work to monitor bots	The technical complexity incurred by bot adoption leads to more work to monitor those bots	P10, P12, P15	And I think for the maintainers, it gives them a little bit more work in some cases, because they have to monitor the setup and make sure everything is going well [P15] Oh, I think they make your they make everything a little bit more complex, right? [...] So I guess it's adding more technical complexity to the project. But what that means is that we can do many more reviews, so usually it works fine, but when it doesn't, it's the perfect more technical. [P12]
		Handling bot failures	Project maintainers need to handle bot failure, due to some bot instabilities	P3, P9, P10, P17, P14	If you rely on these bots and they stop working for some reason then it is a problem. If I automated by releases with semantic-release, and then are some kind of bug in semantic release and I don't know why it basically blocks me to create new releases. Because I automated everything, it is kind hard to manually create the releases.[P3]
ion issues	Bots introducing noise	-	Participants mentioned bots introducing noise	P1, P3, P7, P8, P9, P12, P13, P14, P16, P19, P20, P21	Noise. [P1] Some of them can introduce a lot of noise. [P3] So at some point, they [the bots] might be just too noisy. [P9]
	Ethical issues	Bots with biased behavior	Bot actions and messages might be biased	P3, P9	It is a problem with bots in general. But I could see with you have a bot that works as GreenKeeper in a project and it has some kind of bias built into it, it would be a huge problem. [P3] Yeah, like you there's no criteria to verify the use of you so that the bot is maintained, and is representing the opinion of a specific entity. [P9]
		Bots with malicious intent	Bots with malicious intent might manipulate developers actions	P9, P18	is probably is this good bot or a bad bot because it's about both it could steal our private code, right? [...] It is a bot that actually has malicious intent. [P9] The other option there is, again, related to trust, right, because nobody enforces the bot to disclose. If they are lying for whatever reason, think about it as you can use the bots to manipulate the opinions of developers, right? Like the news, right? They can actually say something that they know is not true, they do it because they will only be very simple as you actually provide a suggestion saying, "Hey, guys, we have a security vulnerability." You know, it's not there, but you're just trying this, you can actually ask a young developer to make something which might introduce a security vulnerability. [P9]
		Bots impersonating developers	Bots migh pretend to be humans developers	P3, P6, P12, P15	The only thing I mentioned is that bots that pretend to be humans can often be met with confusion. So identifying a bot, as a bot, in as many clear ways as possible is really, really important. [P15] This is the confusing thing that we were talking about recently is that it posts the pull requests as like one of the engineers on my team. That's kind of confusing because like it's not always that engineer does it. [P6] Because like, if you're expecting about if you're expecting, like an avatar, like someone to like, interact like a human and then it doesn't I think that's very, that's confusing and like, sets you up for bad expectations. [P6]
		Bots can enforce inflexible rules	Bots can enforce inflexible rules imposed by a community	P3, P7, P15, P17	The social issues are largely down to about being inflexible and not meeting somebody's expectations. [P7] Um, and I've seen a lot of people have problems with that because they say, why is this bot closing my issue that I, you know, so painstakingly opened? And, you know, I shared my thoughts, why is this bot coming along and just, you know, closing my work. [P17] It's not flexible enough and then that combined with our linting and testing suite is too strict. So the biggest complaints we've gotten are that our lint rules and test rule tests are too strict. And of course the bot enforces that. So they'll say things like, wow, the bot is strict. Like, yeah, that's because it's all just doing this thing. "Did you pass the tests? Yes, no." The tests are very strict. So definitely, we've had complaints of saying, "Wow, it's hard to get." It's hard to get a PR from start to finish to pass all the tests and get the bot to accept it. [P7]

Bot interacti	Expectation breakdowns	**For some developers is strange interacting with a bot**	Some developers do not feel comfortable interacting with a bot	P5, P12	"Hey, I'm here to help you" is not super weird. I mean, I think for some people, it's still quite strange, and they're quite surprised by it. [P12] they just say like, receiving thanks from a person is different than thanks from a bot. That's a social aspect.... They just, like, "it's not the same." It feels less sincere to them. [P5]
		Bots can intimidate newcomers	Some newcomers might feel intimidated by bots' interactions and feedback	P2, P12, P17	It can be intimidating, especially if it's the first time you're contributing. [P2] I think there's also, you know, new contributors, I mean to an open source project, interacting with the bot that they've never seen or heard of before, that can be really confusing. And you know, that can be difficult for or intimidating for open source contributors. [P17] because it's not obvious what's happening like I think if you're new to a project then You don't … you might not be expecting bots, Righ? So I think it's, I think it's a sort of ... Yeah, if you don't know to expect it, then that could be confusing. [P12]
	Bot communication issues	**Bots do not contextualize their actions**	Bots do not understand the context of what they are doing	P4	Communication is hard, right? You know, a bot is doing a job and it has to communicate something, whether it's that you have a typo and trying to fix the typo, or your test doesn't pass. It doesn't tell you it. A human can be like, it's not passing because we change this and all these things happen. They don't understand. Let me see... It doesn't understand the context of why something's wrong. You know, a typo. [P4]
		Non-comprehensive feedback	The feedback provided might be unclear, or even poor	P4, P15, P17	Well, you know, something that will comment with information that's unclear. And then you then need to go and like ask a human for clarity. [P17]
		Bots do not provide actionable changes	The feedback provided by bots are not actionable	P8, P9	It's great to see yes or no, but if it's not actionable, then it's not useful for you. [P8] If the answer is like, "Hey, you decreased coverage," or like "you didn't meet some standard." How can you go about fixing it and these are like where you should look first. And so being able to provide that I think is extremely important. [P8]
		Interacting with the bot requires previous knowledge	Understanding the bot requires other technical knowledge not related to the bot	P2, P5, P6, P12	I've seen it even in maybe it's my own bot... people did not know how the auto merging works. But they misuse [the bot]... so allowed it to do is... just write the email to the mailing list saying "this is how it works." [P5] And then so they maybe do a bit more … but basically, some people fail to understand the review process and then also can get frustrated with all the [bot] notifications. [P12]
	Bots being intrusive	**Spamming**	Some bots are designed to spamming repositories	P4, P13, P16	I didn't give [to the bot] explicit authorization to enter the repository, it generates a notification that you did not request, which is the pull request. [P16] [...] people want to have agency, they want to have choice. [...] They want to know that they are being corrected because they asked to be corrected. So when I asked for you to do code coverage, and you tell me I'm failing my code coverage, I asked for that.[P4]
		Modifying commits/pull requests	Some bots can be intrusive, modifying commits and pull requests created by humans	P7, P21	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]

Appendix C

Supplementary Material for Chapter 5

In this appendix, we present the complementary material to replicate the method available in Chapter 5. For more details, refer to our Zenodo repository.[1]

C.1 Design Fiction Guide

Starting the session

- Remembering the terms of the consent form.
- Explaining research objectives.
- Asking permission for recording.
- Sending the link to the fictional story [animated video].

How should the super bot act in the following situations? Please, detail your answers as much as possible.

- When a newcomer submits a contribution to an open-source project.
- When a core developer is working on a priority task. (when a core developer works on something and doesn't want to get interrupted)
- When one or more bots are inflating the pull requests with repetitive or verbose information.
- When a bot performs an unsolicited action on a pull request because of a bug or spam.

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

Followed by (if needed):

- What is the most important feature in this case?
- What should this super bot not do in this case?
- How do you think the super bot would manage the action and information generated by other bots?

Wrap-up questions:

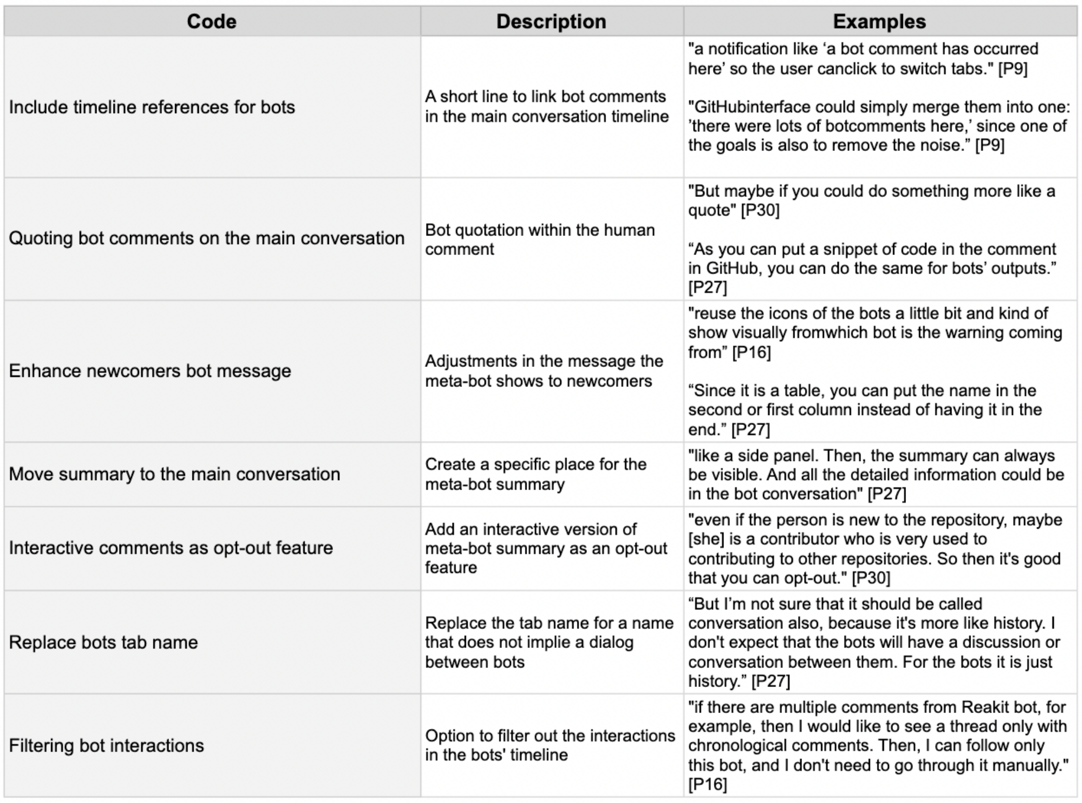
- What else do you think the super bot will be capable of doing to avoid the noise? Give an example.

C.2 Code Book – Designed Strategies

C.2 | CODE BOOK – DESIGNED STRATEGIES

Code		Description	Participants who mentioned	Examples
Information management	Prioritization based on tasks	The super bot presents the most important bots comments based on the task implemented by the developer.	P2, P6, P7, P8, P10, P13, P22, P24, P27	"“it would also be able to sort of filter out what's useful and what's not useful based on that task the developer is actually working on." [P10]
	Prioritization based on issues	The super bot should display the critical bot notifications, including errors and failures.	P2, P13, P15, P18, P21, P23, P24	"if any critical problem happens, then I would like to be notified with a specific bot report, I would receive a critical notification." [P21]
	Summarization of bot comments	The super bot summaries bots outputs into a single comment on a pull request.	P4, P6, P7, P10, P18, P24, P26, P28, P29, P31, P32	"It is difficult to summarize, right? Because, although the message is created by a bot, it's supposedly based on a template." [P31] "Give me a context report or summary. I expect the Super bot to be just one particular comment with some points. Just one comment with everything as a conscious report." [P26]
	Categorization of bot comments	The super bot groups the bot comments based on their types (e.g., testing, security).	P9, P23, P27, P30, P31	" I would categorize the events by type, if it's everything related to coveralls, everything related to license, everything related to vulnerabilities updated versions that we can have a clear vision, because the developer will know what is the priority of each task and wil know if it needs to look at it or not." [P27]
	Aggregating bot comments	The super bot merges bots outputs into a single comment on a pull request.	P4, P8, P10, P14, P15, P16, P24, P30, P32	I mean, the simplest way is just to put them together, but I don't think it's that easy.” [P4]
	Keep the most recent information	The super bot creates a comment and keeps updating the comment with new information from other bots to avoid inflating the pull requests and issues with many comments.	P2, P7, P8, P16, P23, P25, P27	"the super bot just creates one comment and keeps updating it for example.” [P16]
	Interacting with users through natural language	The super bot would provide an interface for communicating with developers in order to understand their requests and answer their questions.	P2, P5, P18, P26	"So in that case, I expect the super bot to be actively involved in the discussion, and not some automated script, which just executes one out of the pre-listed responses." [P26]
	Internationalization	The super bot supports different languages (e.g. German)	P1, P22	"“The language itself could be English or maybe the language you set up on GitHub if there is a feature to say “I'm from Germany. So I want the UI to be in German” than the super bot could use this information and provide German messages.” [P1]
Platform support	Separating bot comments	The GitHub should display bot comments in another panel/session of the pull request interface.	P2, P5, P7, P16, P17, P19, P25, P26, P27, P31, P32	“So I would create another thread for automatic messages. In the pull request, the interface would be able to split the conversation. It would eliminate the problem of typing in the middle of the conversation.” [P17] "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." [P32]
	Bots configuration dashboard	The GitHub would provide a way to access a dashboard that is customizable to the developers needs that could depend on the developers role in the project.	P3, P7, P11, P17	"you end up with tons of repositories and if bots are working on it, you need some overview picture of it." [P3]
Newcomers' assistance	Welcoming message	The super bot welcomes the newcomers by posting a comment, for example "Hi, this is your first contribution. Welcome abroad!"	P2, P10, P15, P20, P21, P23, P24, P25	“as soon as possible you will receive the message “Hi, welcome! I just saw this is your first contribution. Are you aware of the rules of this repository?” or “the rules of this community”. [P2] "to let them know that they're welcome into this community." [P10]
	Explaining rules, instructions, and requirements	The super bot should guide the newcomers and inform them about the projects rules and the requirement to approve the PR.	P2, P6, P13, P14, P15, P20, P21, P23	"[the newcomer] do not understand the rules yet and [the newcomer] don't understand which rules are important." [P13] "the super bot would do an excellent job for a newcomer by explaining why these rules exist." [P13]
	Provide information interactively	The super bot provides information one at a time and communicates with users.	P5, P8, P9, P20, P21	"super bot could do what chatbots do today: guide the new contributor. So, kind of step by step to guide the newcomer" [P5]
	Newcomers pull request notification	The super bot notifies expert developers about a newcomer's contribution.	P1, P6, P8	“[...] and maybe also ping a developer to look at it because it's a new contributor.” [P1]
Notification management	Schedule bot notifications	The super bot would not notify developers on the predefined times they have configured the bot to not interrupt them with notifications (e.g. “do not disturb” mode).	P2, P5, P7, P8, P12, P25, P27	"I was wondering if we could try to create some kind of interface for the level of availability of the interruptions." [P2]
	Do not notify maintainers until the condition is satisfied	The super bot would notify the developers only when the predefined rules were reached.	P16, P27	"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" [P16]
	Notify developers in their idle times	The super bot would not interrupt the developers during critical tasks. Also, the super bot has the ability to learn with the developers’ schedule and adapt to them.	P1, P5, P7, P10, P24, P26, P27	"You could fix this issue by not notifying the developer if you're aware that he is currently working. That's also what other humans would not do." [P1]
	Notify through pre-specified channel	The super bot should send the notifications wherever the developer wants to receive the notification (e.g. email, GitHub notifications, …)	P1, P4, P5, P8, P10, P15, P23, P24	"I think, since it's a super bot, and some of the complaints are that they didn't like information overload, it would be useful for it to make notifications in whatever way the user prefers." [P10]
	Notifying only interested developers	The super bot notifies the developers who are interested in monitoring activities related to a particular Repo/issue/PR.	P2, P16, P25, P27, P28	" if I'm just a contributor, hey, I want to know that notification about that contribution that I made to the project. But as a maintainer, I don't need to know, I don't need to be reminded again, about every contribution that happened when I release a new version of my project" [P25]
Spam and failures management	Prevent repetitive bot activities	The super bot would detect bots that are generating repetitive outcomes and prevent them from acting on pull requests and issues, in order to avoid duplicate messages and spam.	P1, P6, P10, P11, P24, P25, P26, P27	"“If it has the ability to control which bots comment often, then of course, it would be easy to say “no, you only have this one comment and I see that your next comment is exactly the same. I won't let you comment.” That would be maybe the easiest solution. I don't know if ... Okay, assuming that the super bot is super intelligent and really knows that the second comment is a bug. Although it is totally different, it doesn't make sense.” [P1]
	Bugs report	The super bot would be able to create an issue in the smaller bots repository to report bugs with them.	P1, P10	"“Maybe the super bot could create a pull request or issue in the other bots repository and tell them “Hey, look here, I found the bot bugs”. Maybe that was something that's really cool and really futuristic. If super bot creates the issues on other repositories.” [P1]
	Spam messages notification	The super bot would notify developers of repetitive bot messages that might be considered spam.	P1, P10, P25, P26	"“So there should be some sort of feedback for developers whenever they receive a bot notification or so this wouldn't actually be coming from super bot.” [P10]

C.3 Code Book – Improvements



Code	Description	Examples
Include timeline references for bots	A short line to link bot comments in the main conversation timeline	"a notification like 'a bot comment has occurred here' so the user canclick to switch tabs." [P9] "GitHubinterface could simply merge them into one: 'there were lots of botcomments here,' since one of the goals is also to remove the noise." [P9]
Quoting bot comments on the main conversation	Bot quotation within the human comment	"But maybe if you could do something more like a quote" [P30] "As you can put a snippet of code in the comment in GitHub, you can do the same for bots' outputs." [P27]
Enhance newcomers bot message	Adjustments in the message the meta-bot shows to newcomers	"reuse the icons of the bots a little bit and kind of show visually fromwhich bot is the warning coming from" [P16] "Since it is a table, you can put the name in the second or first column instead of having it in the end." [P27]
Move summary to the main conversation	Create a specific place for the meta-bot summary	"like a side panel. Then, the summary can always be visible. And all the detailed information could be in the bot conversation" [P27]
Interactive comments as opt-out feature	Add an interactive version of meta-bot summary as an opt-out feature	"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." [P30]
Replace bots tab name	Replace the tab name for a name that does not implie a dialog between bots	"But I'm not sure that it should be called conversation also, because it's more like history. I don't expect that the bots will have a discussion or conversation between them. For the bots it is just history." [P27]
Filtering bot interactions	Option to filter out the interactions in the bots' timeline	"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." [P16]

References

[Abdellatif *et al.* 2020] Ahmad Abdellatif, Khaled Badran, and Emad Shihab. "Msrbot: using bots to answer questions from software repositories". In: *Empirical Software Engineering* 25.3 (2020), pp. 1834–1863 (cit. on pp. 14, 19, 20).

[Abokhodair *et al.* 2015] Norah Abokhodair, Daisy Yoo, and David W McDonald. "Dissecting a social botnet: growth, content and influence in twitter". In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work Social Computing*. ACM. 2015, pp. 839–851 (cit. on p. 14).

[Aniche *et al.* 2018] Mauricio Aniche *et al.* "How modern news aggregators help development communities shape and share knowledge". In: *ICSE'18*. 2018, pp. 499–510 (cit. on pp. 36, 81).

[Babar *et al.* 2007] Muhammad Ali Babar, Dietmar Winkler, and Stefan Biffl. "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. 2007, pp. 430–439 (cit. on p. 91).

[Balali *et al.* 2018] Sogol Balali, Igor Steinmacher, Umayal Annamalai, Anita Sarma, and Marco Aurelio Gerosa. "Newcomers' barriers... is that all? an analysis of mentors' and newcomers' barriers in oss projects". In: *Computer Supported Cooperative Work (CSCW)* 27.3 (2018), pp. 679–714 (cit. on p. 49).

[Bawden and Robinson 2009] David Bawden and Lyn Robinson. "The dark side of information: overload, anxiety and other paradoxes and pathologies". In: *Journal of information science* 35.2 (2009), pp. 180–191 (cit. on p. 65).

[Benjamini and Hochberg 1995] Yoav Benjamini and Yosef Hochberg. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the Royal statistical society: series B (Methodological)* 57.1 (1995), pp. 289–300 (cit. on p. 33).

[Benotti *et al.* 2014] Luciana Benotti, María Cecilia Martínez, and Fernando Schapachnik. "Engaging high school students using chatbots". In: *Proceedings of the 2014 conference on Innovation technology in computer science education*. ACM. 2014, pp. 63–68 (cit. on p. 14).

[Bernard 2017] Harvey Russell Bernard. *Research methods in anthropology: Qualitative and quantitative approaches.* Rowman & Littlefield, 2017 (cit. on p. 36).

[Beschastnikh *et al.* 2017] Ivan Beschastnikh, Mircea F Lungu, and Yanyan Zhuang. “Accelerating software engineering research adoption with analysis bots”. In: *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER).* IEEE. 2017, pp. 35–38 (cit. on pp. 23, 24).

[Bii 2013] Patrick Bii. “Chatbot technology: a possible means of unlocking student potential to learn how to learn”. In: *Educational Research* 4.2 (2013), pp. 218–221 (cit. on p. 14).

[Bland and Altman 1997] John Martin Bland and Douglas G. Altman. “Statistics notes: cronbach’s alpha”. In: *Bmj* 314.7080 (1997), p. 572 (cit. on p. 93).

[Bleecker 2004] Julian C Bleecker. *The reality effect of technoscience.* University of California, Santa Cruz, 2004 (cit. on p. 80).

[Blythe 2014] Mark Blythe. “Research through design fiction: narrative in real and imaginary abstracts”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM. 2014, pp. 703–712 (cit. on pp. 7, 71, 79).

[Blythe and Encinas 2016] Mark Blythe and Enrique Encinas. “The co-ordinates of design fiction: extrapolation, irony, ambiguity and magic”. In: *Proceedings of the 19th international conference on supporting group work.* ACM. 2016, pp. 345–354 (cit. on p. 79).

[Bradley *et al.* 2018] Nick C. Bradley, Thomas Fritz, and Reid Holmes. “Context-aware conversational developer assistants”. In: *Proceedings of the 40th International Conference on Software Engineering.* ICSE ’18. Gothenburg, Sweden: ACM, 2018, pp. 993–1003. isbn: 978-1-4503-5638-1. doi: 10.1145/3180155.3180238. url: http://doi.acm.org/10.1145/3180155.3180238 (cit. on p. 15).

[Brown and Parnin 2019] Chris Brown and Chris Parnin. “Sorry to bother you: designing bots for effective recommendations”. In: *Proceedings of the 1st International Workshop on Bots in Software Engineering.* BotSE ’19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 54–58. doi: 10.1109/BotSE.2019.00021. url: https://doi.org/10.1109/BotSE.2019.00021 (cit. on pp. 1, 2, 18, 21, 22, 24, 25, 63, 65, 95).

[Cai *et al.* 2019] Liang Cai *et al.* “Answerbot: an answer summary generation tool based on stack overflow”. In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* ACM. 2019, pp. 1134–1138 (cit. on p. 20).

[Candello, Pichiliani, *et al.* 2019] Heloisa Candello, Mauro Pichiliani, Mairieli Wessel, Claudio Pinhanez, and Michael Muller. "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. 2019, p. 15 (cit. on pp. 79, 80, 82).

[Candello, Vasconcelos, *et al.* 2017] Heloisa Candello, Marisa Vasconcelos, and Claudio Pinhanez. "Evaluating the conversation flow and content quality of a multi-bot conversational system". In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2017 (cit. on p. 71).

[Carmines and Zeller 1979] E.G. Carmines and R.A. Zeller. *Reliability and Validity Assessment*. 07 no. 17. SAGE Publications, 1979. isbn: 9780803913714. url: http://books.google.com.br/books?id=BN%5C_MMD9BHogC (cit. on p. 93).

[Carvalho *et al.* 2020] Antonio Carvalho *et al.* "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)*. IEEE. 2020, pp. 161–171 (cit. on pp. 1, 21, 22, 24).

[Cassee *et al.* 2020] Nathan Cassee, Bogdan Vasilescu, and Alexander Serebrenik. "The silent helper: the impact of continuous integration on code reviews". In: *27th IEEE International Conference on Software Analysis, Evolution and Reengineering*. IEEE Computer Society. 2020 (cit. on pp. 28, 32, 35).

[Cerezo *et al.* 2019] Jhonny Cerezo, Juraj Kubelka, Romain Robbes, and Alexandre Bergel. "Building an expert recommender chatbot". In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 59–63. doi: 10.1109/BotSE.2019.00022. url: https://doi.org/10.1109/BotSE.2019.00022 (cit. on pp. 19, 20).

[Charmaz 2006] Kathy Charmaz. *Constructing grounded theory: A practical guide through qualitative analysis*. sage, 2006 (cit. on p. 54).

[Chaves and Marco Aurelio Gerosa 2018] Ana Paula Chaves and Marco Aurelio Gerosa. "Single or multiple conversational agents? an interactional coherence comparison". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–13 (cit. on p. 6).

[Chaves and Marco Aurelio Gerosa 2020] Ana Paula Chaves and Marco Aurelio Gerosa. "How should my chatbot interact? a survey on social characteristics in human–chatbot interaction design". In: *International Journal of Human–Computer Interaction* (2020), pp. 1–30 (cit. on p. 49).

[Chen *et al.* 2012] Hui Hui Chen *et al.* "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*. 2012, H1C-1-H1C–5. doi: 10.1109/TALE.2012.6360324 (cit. on p. 91).

[Cheon and Su 2017] EunJeong Cheon and Norman Makoto Su. “Configuring the user: robots have needs too”. In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. CSCW ’17. Portland, Oregon, USA: ACM, 2017, pp. 191–206. isbn: 978-1-4503-4335-0. doi: 10.1145/2998181.2998329. url: http://doi.acm.org/10.1145/2998181.2998329 (cit. on p. 79).

[Cheon and Su 2018] EunJeong Cheon and Norman Makoto Su. “Futuristic autobiographies: weaving participant narratives to elicit values around robots”. In: *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’18. Chicago, IL, USA: ACM, 2018, pp. 388–397. isbn: 978-1-4503-4953-6. doi: 10.1145/3171221.3171244. url: http://doi.acm.org/10.1145/3171221.3171244 (cit. on p. 79).

[Cook and D. Campbell 1979] Thomas D Cook and D T Campbell. *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. English. Houghton Mifflin, 1979 (cit. on p. 33).

[Cosley *et al.* 2007] Dan Cosley, Dan Frankowski, Loren Terveen, and John Riedl. “Suggestbot: using intelligent task routing to help people find work in wikipedia”. In: *Proceedings of the 12th international conference on Intelligent user interfaces*. ACM. 2007, pp. 32–41 (cit. on p. 14).

[Coulton *et al.* 2017] Paul Coulton, Joseph Lindley, Miriam Sturdee, and Mike Stead. “Design fiction as world building”. In: *Proceedings of Research Through Design Conference 2017*. Mar. 2017 (cit. on p. 79).

[Creswell 2003] John Creswell. “Mixed methods procedures”. In: *Research design: Qualitative, quantitative, and mixed methods approaches* 3 (2003), pp. 203–240 (cit. on pp. 5, 27).

[Dabbish *et al.* 2012] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. “Social coding in github: transparency and collaboration in an open software repository”. In: *CSCW*. New York, NY, USA: ACM, 2012, pp. 1277–1286 (cit. on p. 16).

[Dagli 2019] Meric Dagli. “Designing for Trust”. PhD thesis. figshare, 2019 (cit. on pp. 6, 71, 72).

[Dale 2016] Robert Dale. “The return of the chatbots”. In: *Natural Language Engineering* 22.5 (2016), pp. 811–817 (cit. on pp. 13, 14).

[Davis 1989] Fred D Davis. “Perceived usefulness, perceived ease of use, and user acceptance of information technology”. In: *MIS quarterly* (1989), pp. 319–340 (cit. on pp. 90, 91).

[Dey *et al.* 2020] Tapajit Dey *et al.* “Detecting and characterizing bots that commit code”. In: *Proceedings of the 17th International Conference on Mining Software Repositories*. MSR ’20. Seoul, Republic of Korea: Association for Computing Machinery, 2020, pp. 209–219. isbn: 9781450375177. doi: 10.1145/3379597.3387478. url: https://doi.org/10.1145/3379597.3387478 (cit. on pp. 23, 24).

[Dias *et al.* 2016] L. F. Dias, I. Steinmacher, G. Pinto, D. A. D. Costa, and M. Gerosa. “How does the shift to GitHub impact project collaboration?” In: *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Oct. 2016, pp. 473–477. doi: 10.1109/ICSME.2016.78 (cit. on p. 46).

[Dominic *et al.* 2020] James Dominic, Jada Houser, Igor Steinmacher, Charles Ritter, and Paige Rodeghero. “Conversational bot for newcomers onboarding to open source projects”. In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020, pp. 46–50 (cit. on pp. 19, 20).

[Easterbrook *et al.* 2008] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. “Selecting empirical methods for software engineering research”. In: *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311 (cit. on pp. 5, 27).

[Ellwart *et al.* 2015] Thomas Ellwart, Christian Happ, Andrea Gurtner, and Oliver Rack. “Managing information overload in virtual teams: effects of a structured online team adaptation on cognition and performance”. In: *European Journal of Work and Organizational Psychology* 24.5 (2015), pp. 812–826 (cit. on pp. 49, 67).

[Encinas and Blythe 2016] Enrique Encinas and Mark Blythe. “The solution printer: magic realist design fiction”. In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM. 2016, pp. 387–396 (cit. on p. 79).

[Erlenhov, Neto, *et al.* 2016] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. “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. 2016 (cit. on pp. 18, 49, 65, 67, 95, 99).

[Erlenhov, Oliveira Neto, *et al.* 2019] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Riccardo Scandariato, and Philipp Leitner. “Current and future bots in software development”. In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE ’19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 7–11. doi: 10.1109/BotSE.2019.00009. url: https://doi.org/10.1109/BotSE.2019.00009 (cit. on pp. 15, 16, 20, 66).

[Farooq and Grudin 2016] Umer Farooq and Jonathan Grudin. “Human-computer integration”. In: *interactions* 23.6 (2016), pp. 27–32 (cit. on pp. 13, 15).

[Forlano and Mathew 2014] Laura Forlano and Anijo Mathew. “From design fiction to design friction: speculative and participatory design of values-embedded urban technology”. In: *Journal of Urban Technology* 21.4 (2014), pp. 7–24 (cit. on p. 80).

[Fritsch *et al.* 2013] Jonas Fritsch, Morten Breinbjerg, and Ditte Amund Basballe. “Ekkomaten—exploring the echo as a design fiction concept”. In: *Digital Creativity* 24.1 (2013), pp. 60–74 (cit. on p. 79).

[Fryer *et al.* 2017] Luke K Fryer, Mary Ainley, Andrew Thompson, Aaron Gibson, and Zelinda Sherlock. “Stimulating and sustaining interest in a language course: an experimental comparison of chatbot and human task partners”. In: *Computers in Human Behavior* 75 (2017), pp. 461–468 (cit. on p. 14).

[Gałecki and Burzykowski 2013] Andrzej Gałecki and Tomasz Burzykowski. *Linear mixed-effects models using R: A step-by-step approach.* Springer Science & Business Media, 2013 (cit. on p. 34).

[R Stuart Geiger 2013] R Stuart Geiger. “Are computers merely supporting cooperative work: towards an ethnography of bot development”. In: *Proceedings of the 2013 conference on Computer supported cooperative work companion.* ACM. 2013, pp. 51–56 (cit. on pp. 13, 14).

[R. Stuart Geiger and Halfaker 2017] R. Stuart Geiger and Aaron Halfaker. “Operationalizing conflict and cooperation between automated software agents in wikipedia: a replication and expansion of ’even good bots fight’”. In: *Proc. ACM Hum.-Comput. Interact.* 1.CSCW (Dec. 2017), 49:1–49:33. issn: 2573-0142. doi: 10.1145/3134684. url: http://doi.acm.org/10.1145/3134684 (cit. on p. 14).

[Glaser and Anselm L Strauss 2017] Barney G Glaser and Anselm L Strauss. *Discovery of grounded theory: Strategies for qualitative research.* Routledge, 2017 (cit. on pp. 37, 46, 54, 68, 84, 97).

[Gnewuch *et al.* 2017] Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. “Towards designing cooperative and social conversational agents for customer service”. In: (2017) (cit. on pp. 14, 66).

[Golzadeh *et al.* 2021] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. “A ground-truth dataset and classification model for detecting bots in github issue and pr comments”. In: *Journal of Systems and Software* 175 (2021), p. 110911. issn: 0164-1212. doi: https://doi.org/10.1016/j.jss.2021.110911. url: https://www.sciencedirect.com/science/article/pii/S016412122100008X (cit. on pp. 23, 24, 64).

[Gousios, Pinzger, *et al.* 2014a] Georgios Gousios, Martin Pinzger, and Arie van Deursen. "An exploratory study of the pull-based software development model". In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: ACM, 2014, pp. 345–355. isbn: 978-1-4503-2756-5. doi: 10.1145/2568225.2568260. url: http://doi.acm.org/10.1145/2568225.2568260 (cit. on p. 16).

[Gousios, Pinzger, *et al.* 2014b] Georgios Gousios, Martin Pinzger, and Arie van Deursen. "An exploratory study of the pull-based software development model". In: *Proceedings of the 36th International Conference on Software Engineering*. ACM. 2014, pp. 345–355 (cit. on p. 35).

[Gousios and Spinellis 2012] Georgios Gousios and Diomidis Spinellis. "GHTorrent: GitHub's data from a firehose". In: *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE. 2012, pp. 12–21 (cit. on pp. 28, 34).

[Gousios, Storey, *et al.* 2016] Georgios Gousios, Margaret-Anne Storey, and Alberto Bacchelli. "Work practices and challenges in pull-based development: the contributor's perspective". In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. Austin, Texas: ACM, 2016, pp. 285–296. isbn: 978-1-4503-3900-1. doi: 10.1145/2884781.2884826. url: http://doi.acm.org/10.1145/2884781.2884826 (cit. on pp. 1, 17).

[Goyal *et al.* 2018] Raman Goyal, Gabriel Ferreira, Christian Kästner, and James Herbsleb. "Identifying unusual commits on github". In: *Journal of Software: Evolution and Process* 30.1 (2018), e1893 (cit. on p. 65).

[Harmon *et al.* 2017] Ellie Harmon, Chris Bopp, and Amy Voida. "The design fictions of philanthropic it: stuck between an imperfect present and an impossible future". In: May 2017, pp. 7015–7028. doi: 10.1145/3025453.3025650 (cit. on p. 79).

[Healy 2012] Tim Healy. "The unanticipated consequences of technology". In: *Nanotechnology: ethical and social Implications* (2012), pp. 155–173 (cit. on p. 1).

[Hove and Anda 2005] Siw Elisabeth Hove and Bente Anda. "Experiences from conducting semi-structured interviews in empirical software engineering research". In: *Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS'05)*. IEEE. 2005, 10–pp (cit. on pp. 37, 53, 82).

[Imbens and Lemieux 2008] Guido W Imbens and Thomas Lemieux. "Regression discontinuity designs: a guide to practice". In: *Journal of econometrics* 142.2 (2008), pp. 615–635 (cit. on p. 32).

[Iqbal and Horvitz 2010] Shamsi T Iqbal and Eric Horvitz. "Notifications and awareness: a field study of alert usage and preferences". In: *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. 2010, pp. 27–30 (cit. on pp. 65, 67).

[Jain *et al.* 2018] Mohit Jain, Ramachandra Kota, Pratyush Kumar, and Shwetak N Patel. “Convey: exploring the use of a context view for chatbots”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM. 2018, p. 468 (cit. on p. 14).

[Jeffery *et al.* 2017] Alvin D Jeffery, Laurie L Novak, Betsy Kennedy, Mary S Dietrich, and Lorraine C Mion. “Participatory design of probability-based decision support tools for in-hospital nurses”. In: *Journal of the American Medical Informatics Association* 24.6 (2017), pp. 1102–1110 (cit. on p. 90).

[Jones *et al.* 2004] Quentin Jones, Gilad Ravid, and Sheizaf Rafaeli. “Information overload and the message dynamics of online interaction spaces: a theoretical model and empirical exploration”. In: *Information systems research* 15.2 (2004), pp. 194–210 (cit. on p. 65).

[Kalliamvakou *et al.* 2014] Eirini Kalliamvakou *et al.* “The promises and perils of mining GitHub”. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. MSR 2014. Hyderabad, India: ACM, 2014, pp. 92–101. isbn: 978-1-4503-2863-0. doi: 10.1145/2597073.2597074. url: http://doi.acm.org/10.1145/2597073.2597074 (cit. on p. 46).

[D. Kavaler *et al.* 2019] D. Kavaler, A. Trockman, B. Vasilescu, and V. Filkov. “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)*. 2019, pp. 476–487 (cit. on p. 1).

[David Kavaler *et al.* 2019] David Kavaler, Asher Trockman, Bogdan Vasilescu, and Vladimir Filkov. “Tool choice matters: JavaScript quality assurance tools and usage outcomes in GitHub projects”. In: *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press. 2019, pp. 476–487 (cit. on p. 28).

[Kerry *et al.* 2008] Alice Kerry, Richard Ellis, and Susan Bull. “Conversational agents in e-learning”. In: *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer. 2008, pp. 169–182 (cit. on p. 14).

[Kim *et al.* 2007] Hyekyung Kim, Miguel E Ruiz, and Lorna Peterson. “Usability and effectiveness evaluation of a course-advising chat bot”. In: *Proceedings of the American Society for Information Science and Technology* 44.1 (2007), pp. 1–5 (cit. on p. 14).

[Kinsman *et al.* 2021] Timothy Kinsman, Mairieli Wessel, Marco Gerosa, and Christoph Treude. “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. 2021 (cit. on p. 66).

[Kirby 2010] David Kirby. “The future is now: diegetic prototypes and the role of popular films in generating real-world technological development”. In: *Social Studies of Science* 40.1 (2010), pp. 41–70 (cit. on p. 80).

[Kirman *et al.* 2013] Ben Kirman, Conor Linehan, Shaun Lawson, and Dan O'Hara. "Chi and the future robot enslavement of humankind: a retrospective". In: *CHI'13 Extended Abstracts on Human Factors in Computing Systems.* 2013, pp. 2199–2208 (cit. on p. 79).

[Knutz and Markussen 2014] Eva Knutz and Thomas Markussen. "The role of fiction in experiments within design, art & architecture-towards a new typology of design fiction". In: *Artifact: Journal of Design Practice* 3.2 (2014), pp. 8–1 (cit. on p. 95).

[Kuznetsova *et al.* 2017] Alexandra Kuznetsova, Per B Brockhoff, and Rune Haubo Bojesen Christensen. "Lmertest package: tests in linear mixed effects models". In: *Journal of Statistical Software* 82.13 (2017) (cit. on p. 34).

[Latham *et al.* 2010] Annabel M Latham, Keeley A Crockett, David A McLean, Bruce Edmonds, and Karen O'Shea. "Oscar: an intelligent conversational agent tutor to estimate learning styles". In: *International Conference on Fuzzy Systems.* Washington, DC, USA: IEEE, 2010, pp. 1–8 (cit. on p. 96).

[C. Lebeuf, Storey, *et al.* 2018] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. "Software bots". In: *IEEE Software* 35.1 (2018), pp. 18–23 (cit. on pp. 16, 63).

[C. Lebeuf, Zagalsky, *et al.* 2019] Carlene Lebeuf, Alexey Zagalsky, Matthieu Foucault, and Margaret-Anne Storey. "Defining and classifying software bots: a faceted taxonomy". In: *Proceedings of the 1st International Workshop on Bots in Software Engineering.* BotSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 1–6. doi: 10.1109/BotSE.2019.00008. url: https://doi.org/10.1109/BotSE.2019.00008 (cit. on pp. 13, 14, 16).

[C. R. Lebeuf 2018] Carlene R Lebeuf. "A taxonomy of software bots: towards a deeper understanding of software bot characteristics". PhD thesis. 2018 (cit. on pp. 13–15, 20).

[Lee *et al.* 2017] Minha Lee, Lily Frank, Femke Beute, Yvonne de Kort, and Wijnand IJsselsteijn. "Bots mind the social-technical gap". In: *Proceedings of 15th European conference on computer-supported cooperative work-exploratory papers.* European Society for Socially Embedded Technologies (EUSSET). 2017 (cit. on p. 13).

[Lin *et al.* 2016] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. "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. 2016, pp. 333–336 (cit. on pp. 16, 18, 20).

[Lindley *et al.* 2016] Joseph Lindley, Paul Coulton, and Emmett L Brown. "Peer review and design fiction: "honestly, they're not just made up."" In: *CHI Extended Abstracts (Alt. CHI). ACM* (2016) (cit. on p. 79).

[Linehan *et al.* 2014] Conor Linehan *et al.* “Alternate endings: using fiction to explore design futures”. In: *CHI’14 Extended Abstracts on Human Factors in Computing Systems*. ACM. 2014, pp. 45–48 (cit. on p. 79).

[Liu *et al.* 2020] Dongyu Liu, Micah J. Smith, and Kalyan Veeramachaneni. “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*. CHI EA ’20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–8. isbn: 9781450368193. doi: 10.1145/3334480.3382998. url: https://doi.org/10.1145/3334480.3382998 (cit. on pp. 2, 49, 63, 66, 67).

[Long *et al.* 2017] Kiel Long *et al.* “Could you define that in bot terms?: requesting, creating and using bots on reddit”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM. 2017, pp. 3488–3500 (cit. on p. 14).

[López and Guerrero 2016] Gustavo López and Luis A Guerrero. “Ubiquitous notification mechanism to provide user awareness”. In: *Advances in Ergonomics in Design*. Springer, 2016, pp. 689–700 (cit. on p. 65).

[Lupton 2017] Ellen Lupton. *Design is storytelling*. 2017 (cit. on p. 79).

[Matthies *et al.* 2019] Christoph Matthies, Franziska Dobrigkeit, and Guenter Hesse. “An additional set of (automated) eyes: chatbots for agile retrospectives”. In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE ’19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 34–37. doi: 10.1109/BotSE.2019.00017. url: https://doi.org/10.1109/BotSE.2019.00017 (cit. on pp. 14, 18, 20).

[Maurer and Weihe 2015] Daniel Maurer and Karsten Weihe. “Benjamin franklin’s decision method is acceptable and helpful with a conversational agent”. In: *Intelligent Interactive Multimedia Systems and Services*. Springer, 2015, pp. 109–120 (cit. on p. 14).

[Maus 2017] Gregory Maus. “A typology of socialbots (abbrev.)” In: *Proceedings of the 2017 ACM on Web Science Conference*. ACM. 2017, pp. 399–400 (cit. on p. 15).

[McIntosh *et al.* 2014] Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E Hassan. “The impact of code review coverage and code review participation on software quality: a case study of the qt, vtk, and itk projects”. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. 2014, pp. 192–201 (cit. on p. 1).

[Mendez *et al.* 2018] C. Mendez *et al.* “Open source barriers to entry, revisited: a sociotechnical perspective”. In: *Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. 2018, pp. 1004–1015 (cit. on pp. 49, 64).

[Merriam 1998] Sharan B Merriam. *Qualitative Research and Case Study Applications in Education. Revised and Expanded from" Case Study Research in Education.".* ERIC, 1998 (cit. on pp. 54, 90).

[Miller 1956] George A Miller. "The magical number seven, plus or minus two: some limits on our capacity for processing information." In: *Psychological review* 63.2 (1956), p. 81 (cit. on pp. 65, 95).

[Mimoun *et al.* 2017] Mohammed Slim Ben Mimoun, Ingrid Poncin, and Marion Garnier. "Animated conversational agents and e-consumer productivity: the roles of agents and individual characteristics". In: *Information Management* 54.5 (2017), pp. 545–559 (cit. on p. 14).

[Mirhosseini and Parnin 2017] Samim Mirhosseini and Chris Parnin. "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*. ASE 2017. Urbana-Champaign, IL, USA: IEEE Press, 2017, pp. 84–94. isbn: 978-1-5386-2684-9. url: http://dl.acm.org/citation.cfm?id=3155562.3155577 (cit. on pp. 1, 2, 21–25, 65, 95).

[Monperrus 2019] Martin Monperrus. "Explainable software bot contributions: case study of automated bug fixes". In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 12–15. doi: 10.1109/BotSE.2019.00010. url: https://doi.org/10.1109/BotSE.2019.00010 (cit. on pp. 1, 2, 18, 21).

[Monperrus, Urli, *et al.* 2018] Martin Monperrus, Simon Urli, *et al.* "Human-competitive patches in automatic program repair with repairnator". In: *CoRR* abs/1810.05806 (2018). arXiv: 1810.05806. url: http://arxiv.org/abs/1810.05806 (cit. on pp. 21, 24).

[Monperrus, Urli, *et al.* 2019] Martin Monperrus, Simon Urli, *et al.* "Repairnator patches programs automatically". In: *Ubiquity* 2019.July (July 2019), 2:1–2:12. issn: 1530-2180. doi: 10.1145/3349589. url: http://doi.acm.org/10.1145/3349589 (cit. on pp. 15, 24).

[Mulder 2013] KF Mulder. "Impact of new technologies: how to assess the intended and unintended effects of new technologies". In: *Handb. Sustain. Eng.(2013)* (2013) (cit. on p. 2).

[M. Muller and Erickson 2018] Michael Muller and Thomas Erickson. "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*. CHI EA '18. Montreal QC, Canada: ACM, 2018, alt14:1–alt14:10. isbn: 978-1-4503-5621-3. doi: 10.1145/3170427.3188407. url: http://doi.acm.org/10.1145/3170427.3188407 (cit. on pp. 79, 82).

[M. Muller and Liao 2017] Michael Muller and Q Vera Liao. “Exploring ai ethics and values through participatory design fictions”. In: *Human Computer Interaction Consortium* (2017). URL: https://www.slideshare.net/traincroft/hcic-muller-and-liao-participatory-design-fictions-77345391 (cit. on p. 79).

[M. J. Muller 2007] Michael J Muller. “Participatory design: the third space in hci”. In: *The human-computer interaction handbook*. CRC press, 2007, pp. 1087–1108 (cit. on p. 80).

[Müller-Birn *et al.* 2013] Claudia Müller-Birn, Leonhard Dobusch, and James D Herbsleb. “Work-to-rule: the emergence of algorithmic governance in wikipedia”. In: *Proceedings of the 6th International Conference on Communities and Technologies*. ACM. 2013, pp. 80–89 (cit. on pp. 18, 64, 66).

[Murgia *et al.* 2016] Alessandro Murgia, Daan Janssens, Serge Demeyer, and Bogdan Vasilescu. “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*. ACM. 2016, pp. 1272–1279 (cit. on pp. 19, 20, 64).

[Nägele *et al.* 2018] Larissa Vivian Nägele, Merja Ryöppy, and Danielle Wilde. “Pdfi: participatory design fiction with vulnerable users”. In: *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*. 2018, pp. 819–831 (cit. on p. 80).

[Nakagawa and Schielzeth 2013] Shinichi Nakagawa and Holger Schielzeth. “A general and simple method for obtaining r2 from generalized linear mixed-effects models”. In: *Methods in ecology and evolution* 4.2 (2013), pp. 133–142 (cit. on p. 34).

[Nakamura *et al.* 2012] Kazuaki Nakamura, Koh Kakusho, Tetsuo Shoji, and Michihiko Minoh. “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*. Berlin, Heidelberg: Springer, 2012, pp. 425–433 (cit. on p. 96).

[Nematzadeh *et al.* 2016] Azadeh Nematzadeh, Giovanni Luca Ciampaglia, Yong-Yeol Ahn, and Alessandro Flammini. “Information overload in group communication: from conversation to cacophony in the twitch chat”. In: *Royal Society open science* 6.10 (2016), p. 191412 (cit. on pp. 2, 65).

[Noortman *et al.* 2019] Renee Noortman, Britta F. Schulte, Paul Marshall, Saskia Bakker, and Anna L. Cox. “Hawkeye - deploying a design fiction probe”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI ’19. Glasgow, Scotland Uk: ACM, 2019, 422:1–422:14. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300652. URL: http://doi.acm.org/10.1145/3290605.3300652 (cit. on p. 79).

[Paikari, Choi, *et al.* 2019] Elahe Paikari, JaeEun Choi, *et al.* "A chatbot for conflict detection and resolution". In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 29–33. doi: 10.1109/BotSE.2019.00016. url: https://doi.org/10.1109/BotSE.2019.00016 (cit. on pp. 15, 19, 20).

[Paikari and Hoek 2018] Elahe Paikari and Andre van der Hoek. "A framework for understanding chatbots and their future". In: *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*. CHASE '18. Gothenburg, Sweden: ACM, 2018, pp. 13–16. isbn: 978-1-4503-5725-8. doi: 10.1145/3195836.3195859. url: http://doi.acm.org/10.1145/3195836.3195859 (cit. on pp. 2, 63).

[Patton 2014] Michael Quinn Patton. *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications, 2014 (cit. on pp. 37, 53, 84).

[Peng and Ma 2019] Zhenhui Peng and Xiaojuan Ma. "Exploring how software developers work with mention bot in github". In: *CCF Transactions on Pervasive Computing and Interaction* 1.3 (Nov. 2019), pp. 190–203. issn: 2524-5228. doi: 10.1007/s42486-019-00013-2. url: https://doi.org/10.1007/s42486-019-00013-2 (cit. on pp. 22–25, 62, 65, 95).

[Peng, Yoo, *et al.* 2018] Zhenhui Peng, Jeehoon Yoo, Meng Xia, Sunghun Kim, and Xiaojuan Ma. "Exploring how software developers work with mention bot in github". In: *Proceedings of the Sixth International Symposium of Chinese CHI*. ChineseCHI '18. Montreal, QC, Canada: ACM, 2018, pp. 152–155. isbn: 978-1-4503-6508-6. doi: 10.1145/3202667.3202694. url: http://doi.acm.org/10.1145/3202667.3202694 (cit. on pp. 21, 24, 62, 65).

[Pereira 2016] Juanan Pereira. "Leveraging chatbots to improve self-guided learning through conversational quizzes". In: *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*. ACM. 2016, pp. 911–918 (cit. on p. 14).

[Pérez-Soler *et al.* 2017] Sara Pérez-Soler, Esther Guerra, Juan de Lara, and Francisco Jurado. "The rise of the (modelling) bots: towards assisted modelling via social networks". In: *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. ASE 2017. Urbana-Champaign, IL, USA: IEEE Press, 2017, pp. 723–728. isbn: 978-1-5386-2684-9. url: http://dl.acm.org/citation.cfm?id=3155562.3155652 (cit. on p. 16).

[A. Pinheiro *et al.* 2019] André Pinheiro, Caio Rabello, Leonardo Furtado, Gustavo Pinto, and Cleidson Souza. "How do bot developers perceive bot development? a survey". In: *Anais do IV Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software*. SBC. 2019, pp. 11–20 (cit. on p. 20).

[A. M. Pinheiro *et al.* 2019] André M Pinheiro, Caio S Rabello, Leonardo B Furtado, Gustavo Pinto, and Cleidson RB de Souza. “Expecting the unexpected: distilling bot development, challenges, and motivations”. In: *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press. 2019, pp. 51–52 (cit. on p. 20).

[Pinto *et al.* 2017] Gustavo Henrique Lima Pinto, Fernando Figueira Filho, Igor Steinmacher, and Marco Aurelio Gerosa. “Training software engineers using open-source software: the professors’ perspective”. In: *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. 2017, pp. 117–121 (cit. on p. 49).

[Ren *et al.* 2019] Luyao Ren, Shurui Zhou, Christian Kästner, and Andrzej Wąsowski. “Identifying redundancies in fork-based development”. In: *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2019, pp. 230–241 (cit. on pp. 21, 22, 24).

[Romano *et al.* 2006] Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, and Jeff Skowronek. “Appropriate statistics for ordinal level data: should we really be using t-test and cohen’sd for evaluating group differences on the nsse and other surveys”. In: *annual meeting of the Florida Association of Institutional Research*. 2006, pp. 1–33 (cit. on p. 28).

[Romero *et al.* 2020] Ricardo Romero, Esteban Parra, and Sonia Haiduc. “Experiences building an answer bot for gitter”. In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020, pp. 66–70 (cit. on pp. 19, 20).

[Runeson and Höst 2009] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical software engineering* 14.2 (2009), p. 131 (cit. on p. 27).

[Sadeddin *et al.* 2007] Khaled W Sadeddin, Alexander Serenko, and James Hayes. “Online shopping bots for electronic commerce: the comparison of functionality and performance”. In: *International Journal of Electronic Business* 5.6 (2007), p. 576 (cit. on pp. 6, 71, 72).

[Savage *et al.* 2016] Saiph Savage, Andres Monroy-Hernandez, and Tobias Höllerer. “Botivist: calling volunteers to action using online bots”. In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. New York, NY, USA: ACM, 2016, pp. 813–822 (cit. on p. 14).

[Shannon 2001] C. E. Shannon. “A mathematical theory of communication”. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 5.1 (Jan. 2001), pp. 3–55. issn: 1559-1662. doi: 10.1145/584091.584093. url: https://doi.org/10.1145/584091.584093 (cit. on p. 64).

[Shawar and Atwell 2007] Bayan Abu Shawar and Eric Atwell. "Chatbots: are they really useful?" In: *Ldv forum*. Vol. 22. 1. 2007, pp. 29–49 (cit. on p. 13).

[Sheather 2009] Simon Sheather. *A modern approach to regression with R*. Springer Science & Business Media, 2009 (cit. on p. 34).

[Simone *et al.* 1995] Carla Simone, Monica Divitini, and Kjeld Schmidt. "A notation for malleable and interoperable coordination mechanisms for cscw systems". In: *Proceedings of conference on Organizational computing systems*. 1995, pp. 44–54 (cit. on p. 65).

[Singer *et al.* 2014] Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. "Software engineering at the speed of light: how developers stay current using Twitter". In: *36th ICSE*. 2014, pp. 211–221 (cit. on pp. 36, 81).

[Statt 2016] Nick Statt. "Why google's fancy new ai assistant is just called'google'". In: *Retrieved March* 21 (2016), p. 2017 (cit. on p. 14).

[I. Steinmacher, Chaves, *et al.* 2013] Igor Steinmacher, Ana Paula Chaves, and Marco Aurélio Gerosa. "Awareness support in distributed software development: a systematic review and mapping of the literature". In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW)* 22.2-3 (2013), pp. 113–158 (cit. on pp. 64, 65).

[I. Steinmacher, T. Conte, *et al.* 2015] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. "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*. 2015, pp. 1379–1392 (cit. on pp. 2, 49).

[I. Steinmacher, T. U. Conte, *et al.* 2016] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurelio Gerosa. "Overcoming open source project entry barriers with a portal for newcomers". In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. Austin, Texas: ACM, 2016, pp. 273–284. isbn: 978-1-4503-3900-1. doi: 10.1145/2884781.2884806. url: http://doi.acm.org/10.1145/2884781.2884806 (cit. on pp. 2, 91).

[I. Steinmacher, I. Wiese, *et al.* 2013] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, and Marco Aurelio Gerosa. "Why do newcomers abandon open source software projects?" In: *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE. 2013, pp. 25–32 (cit. on p. 1).

[I. F. Steinmacher 2015] Igor Fábio Steinmacher. "Supporting newcomers to overcome the barriers to contribute to open source software projects". PhD thesis. Universidade de São Paulo, 2015 (cit. on p. 16).

[Sterling 2009] Bruce Sterling. "Cover story design fiction". In: *interactions* 16.3 (2009), pp. 20–24 (cit. on p. 79).

[Stol *et al.* 2016] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. "Grounded theory in software engineering research: a critical review and guidelines". In: *Proceedings of the 38th International Conference on Software Engineering*. 2016, pp. 120–131 (cit. on pp. 37, 53).

[Storey, Serebrenik, *et al.* 2020] Margaret-Anne Storey, Alexander Serebrenik, Carolyn Penstein Rosé, Thomas Zimmermann, and James D. Herbsleb. "BOTse: Bots in Software Engineering (Dagstuhl Seminar 19471)". In: *Dagstuhl Reports* 9.11 (2020), pp. 84–96. issn: 2192-5283 (cit. on pp. 2, 63, 64, 66).

[Storey, Treude, *et al.* 2010] Margaret-Anne Storey, Christoph Treude, Arie van Deursen, and Li-Te Cheng. "The impact of social media on software engineering practices and tools". In: *FSE/SDP workshop on Future of Softw Eng Research*. 2010, pp. 359–364 (cit. on pp. 36, 81).

[Storey and Zagalsky 2016] Margaret-Anne Storey and Alexey Zagalsky. "Disrupting developer productivity one bot at a time". In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2016. Seattle, WA, USA: ACM, 2016, pp. 928–931. isbn: 978-1-4503-4218-6. doi: 10.1145/2950290.2983989. url: http://doi.acm.org/10.1145/2950290.2983989 (cit. on pp. 1, 2, 14–16, 18, 47, 63).

[A. L. Strauss and J. M. Corbin 1998] A. L. Strauss and J. M. Corbin. "Basics of qualitative research: techniques and procedures for developing grounded theory sage publications". In: *SAGE Publications* (1998) (cit. on pp. 37, 53, 83).

[A. Strauss and Juliet M Corbin 1997] Anselm Strauss and Juliet M Corbin. *Grounded theory in practice*. Sage, 1997 (cit. on pp. 36, 47, 68, 97).

[Tamayo-Moreno and Pérez-Marín 2017] Silvia Tamayo-Moreno and Diana Pérez-Marín. "Designing and evaluating pedagogic conversational agents to teach children". In: *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering* 11.3 (2017), pp. 488–493 (cit. on p. 14).

[Tegos and Demetriadis 2017] Stergios Tegos and Stavros Demetriadis. "Conversational agents improve peer learning through building on prior knowledge". In: *Journal of Educational Technology Society* 20.1 (2017), pp. 99–111 (cit. on p. 14).

[Thistlethwaite and D. T. Campbell 1960] Donald L Thistlethwaite and Donald T Campbell. "Regression-discontinuity analysis: an alternative to the ex post facto experiment." In: *Journal of Educational psychology* 51.6 (1960), p. 309 (cit. on pp. 5, 27, 32).

[Thomas 2016] NT Thomas. "An e-business chatbot using aiml and lsa". In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2016, pp. 2740–2742 (cit. on p. 14).

[Tonder and Goues 2019] Rijnard van Tonder and Claire Le Goues. “Towards s/engineer/bot: principles for program repair bots”. In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE ’19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 43–47. doi: 10.1109/BotSE.2019.00019. url: https://doi.org/10.1109/BotSE.2019.00019 (cit. on pp. 21, 24).

[Tsay *et al.* 2014] Jason Tsay, Laura Dabbish, and James Herbsleb. “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*. FSE 2014. Hong Kong, China: ACM, 2014, pp. 144–154. isbn: 978-1-4503-3056-5. doi: 10.1145/2635868.2635882. url: http://doi.acm.org/10.1145/2635868.2635882 (cit. on p. 1).

[Turing 1950] Alan M Turing. “Computing machinery and intelligence”. In: *Mind* 59.236 (1950), pp. 433–460 (cit. on p. 13).

[Urli *et al.* 2018] Simon Urli, Zhongxing Yu, Lionel Seinturier, and Martin Monperrus. “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*. ACM. 2018, pp. 95–104 (cit. on pp. 1, 21, 24).

[Vasilescu *et al.* 2015] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. “Quality and productivity outcomes relating to continuous integration in github”. In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. Bergamo, Italy: Association for Computing Machinery, 2015, pp. 805–816. isbn: 9781450336758. doi: 10.1145/2786805.2786850. url: https://doi.org/10.1145/2786805.2786850 (cit. on p. 1).

[Vinciarelli *et al.* 2015] Alessandro Vinciarelli *et al.* “Open challenges in modelling, analysis and synthesis of human behaviour in human–human and human–machine interactions”. In: *Cognitive Computation* 7.4 (2015), pp. 397–413 (cit. on p. 13).

[Vorvoreanu *et al.* 2019] Mihaela Vorvoreanu *et al.* “From gender biases to gender-inclusive design: an empirical investigation”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI ’19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–14. isbn: 9781450359702. doi: 10.1145/3290605.3300283. url: https://doi.org/10.1145/3290605.3300283 (cit. on pp. 49, 64, 96).

[Weizenbaum *et al.* 1966] Joseph Weizenbaum *et al.* “Eliza—a computer program for the study of natural language communication between man and machine”. In: *Communications of the ACM* 9.1 (1966), pp. 36–45 (cit. on p. 13).

[Wessel, Serebrenik, I. Wiese, I. Steinmacher, and Marco A Gerosa 2021] Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco A Gerosa. "Quality gatekeepers: investigating the effects ofcode review bots on pull request activities". In: *EMSE* (2021) (cit. on p. 27).

[Wessel, Serebrenik, I. Wiese, I. Steinmacher, and Marco Aurelio Gerosa 2020] Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco Aurelio Gerosa. "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*. Oct. 2020 (cit. on pp. 2, 48, 64, 99).

[Wessel, Serebrenik, I. S. Wiese, *et al.* 2020] Mairieli Wessel, Alexander Serebrenik, Igor Scaliante Wiese, Igor Steinmacher, and Marco Aurelio Gerosa. "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. 2020 (cit. on pp. 2, 27).

[Wessel, Souza, *et al.* 2018] Mairieli Wessel, Bruno Mendes de Souza, *et al.* "The power of bots: characterizing and understanding bots in oss projects". In: *Proc. ACM Hum.-Comput. Interact.* 2.CSCW (Nov. 2018), 182:1–182:19. issn: 2573-0142. doi: 10.1145/3274451. url: http://doi.acm.org/10.1145/3274451 (cit. on pp. 1, 5, 21, 34, 46, 47, 63, 64, 67, 95, 99).

[Wessel and I. Steinmacher 2020] Mairieli Wessel and Igor Steinmacher. "The inconvenient side of software bots on pull requests". In: *Proceedings of the 2nd International Workshop on Bots in Software Engineering*. BotSE. 2020. doi: 10.1145/3387940.3391504 (cit. on pp. 6, 67).

[Wessel, I. Steinmacher, *et al.* 2019] Mairieli Wessel, Igor Steinmacher, Igor Wiese, and Marco A. Gerosa. "Should i stale or should i close?: an analysis of a bot that closes abandoned issues and pull requests". In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 38–42. doi: 10.1109/BotSE.2019.00018. url: https://doi.org/10.1109/BotSE.2019.00018 (cit. on pp. 5, 18, 21).

[Wessel, I. Wiese, *et al.* 2021] Mairieli Wessel, Igor Wiese, Igor Steinmacher, and Marco A. Gerosa. "Don't disturb me: challenges of interacting with software bots on open source software projects". In: *Proceedings of ACM Human-Computer Interaction* CSCW (2021) (cit. on pp. 6, 51, 80, 83, 95, 96).

[Wilks 2011] Daniel Wilks. *Statistical Methods in the Atmospheric Sciences*. Academic Press. Academic Press, 2011. isbn: 9780123850225 (cit. on p. 28).

[Winarsky *et al.* 2012] Norman Winarsky, Bill Mark, and Henry Kressel. "The development of siri and the sri venture creation process". In: *SRI International, Menlo Park, USA, Tech. Rep* (2012) (cit. on p. 14).

[Wohlin *et al.* 2012] Claes Wohlin *et al. Experimentation in software engineering.* Springer Science & Business Media, 2012 (cit. on p. 46).

[Woods and Patterson 2001] David D Woods and Emily S Patterson. "How unexpected events produce an escalation of cognitive and coordinative demands". In: *PA Hancock, & PA Desmond, Stress, workload, and fatigue. Mahwah, NJ: L. Erlbaum* (2001) (cit. on p. 1).

[Wyrich and Bogner 2019] Marvin Wyrich and Justus Bogner. "Towards an autonomous bot for automatic source code refactoring". In: *Proceedings of the 1st International Workshop on Bots in Software Engineering*. BotSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 24–28. doi: 10.1109/BotSE.2019.00015. url: https://doi.org/10.1109/BotSE.2019.00015 (cit. on pp. 1, 14, 17, 21, 24).

[Bin Xu *et al.* 2014] Bin Xu, Tina Chien-Wen Yuan, Susan R. Fussell, and Dan Cosley. "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 &#38; Social Computing*. CSCW Companion '14. Baltimore, Maryland, USA: ACM, 2014, pp. 41–44. isbn: 978-1-4503-2541-7. doi: 10.1145/2556420.2556789. url: http://doi.acm.org/10.1145/2556420.2556789 (cit. on p. 14).

[Bowen Xu *et al.* 2017] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. "Answerbot: automated generation of answer summary to developersź technical questions". In: *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press. 2017, pp. 706–716 (cit. on pp. 18, 20).

[Yin 2003] Robert K. Yin. "Design and methods". In: *Case study research* 3 (2003) (cit. on p. 27).

[Yu *et al.* 2015] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu. "Wait for it: determinants of pull request evaluation latency on GitHub". In: *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. May 2015, pp. 367–371. doi: 10.1109/MSR.2015.42 (cit. on p. 1).

[Zhao *et al.* 2017] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. "The impact of continuous integration on other software development practices: a large-scale empirical study". In: *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press. 2017, pp. 60–71 (cit. on pp. 28, 32, 34, 35).

[Zheng *et al.* 2018] L Zheng, Christopher M Albano, and Jeffrey V Nickerson. "Steps toward understanding the design and evaluation spaces of bot and human knowledge production systems". In: *Wiki Workshop'19*. 2018 (cit. on p. 18).

[Zimmermann 2016] Thomas Zimmermann. "Card-sorting: from text to themes". In: *Perspectives on Data Science for Software Engineering*. Elsevier, 2016, pp. 137–141 (cit. on p. 76).

[Zue and Glass 2000] Victor W Zue and James R Glass. “Conversational interfaces: advances and challenges”. In: *Proceedings of the IEEE* 88.8 (2000), pp. 1166–1180 (cit. on p. 13).

Index