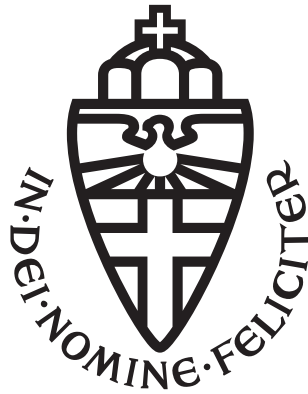


# BACHELOR'S THESIS COMPUTING SCIENCE



RADBOUD UNIVERSITY NIJMEGEN

---

## An Accompaniment Chatbot for People Onboarding to Open-Source Software Projects

---

*Author:*  
Maikel Jans

*First supervisor/assessor:*  
dr. Mairieli Santos Wessel

*Second assessor:*  
dr. Bin Lin

August 21, 2023

## **Abstract**

In this paper, we target the problem of newcomers encountering barriers, created by rules that differ from project to project, that might discourage or hinder their active participation and contributions to open-source software (OSS) projects. This research is intended to enhance the onboarding process for newcomers in open-source software projects by introducing a chatbot designed to assist with understanding the guidelines of submitting their contributions. The user study is centered around a specific repository, chosen as a case study, and utilizes the Technology Acceptance Model to evaluate the chatbot's perceived usefulness and ease of use. Through quantitative analysis, we assessed the participants' perceptions of the chatbot. Qualitative insights were gathered through open-ended feedback. The findings suggest that while the chatbot was generally well-received, opportunities exist for improving its responsiveness and adaptability. The results underscore the potential for chatbots to serve as valuable tools for facilitating newcomers' engagement in OSS projects, albeit with further improvements.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Onboarding to Open Source Projects . . . . .	5
2.2	Chatbots . . . . .	7
2.3	Chatbot Development . . . . .	8
2.3.1	Rasa . . . . .	8
2.3.2	Botfront . . . . .	9
<b>3</b>	<b>Related Work</b>	<b>10</b>
3.1	Bots in Software Engineering . . . . .	10
3.2	Onboarding to OSS projects . . . . .	11
<b>4</b>	<b>Methodology</b>	<b>13</b>
4.1	Chatbot Context . . . . .	13
4.2	Chatbot Implementation Details . . . . .	15
4.2.1	Website Hosting Chatbot . . . . .	17
4.3	Questionnaire . . . . .	18
<b>5</b>	<b>Results</b>	<b>21</b>

5.1	Quantitative Analysis . . . . .	21
5.2	Qualitative Analysis of the Open-Questions . . . . .	24
<b>6</b>	<b>Discussion</b>	<b>28</b>
6.1	Limitations . . . . .	30
6.1.1	Chatbot Implementation Limitations . . . . .	30
6.1.2	Questionnaire Limitations . . . . .	32
<b>7</b>	<b>Conclusions</b>	<b>33</b>
<b>A</b>	<b>Appendix</b>	<b>37</b>

# Chapter 1

## Introduction

When developing software in a distributed and collaborative environment, developers will almost certainly use version control tools to collaborate. With GitHub<sup>1</sup> being one of the most popular web-based hosting service for software development and version control [3]. The ability to create pull requests (PRs) is an important feature of GitHub. They are used for submitting contributions to the code base as well as for quality assurance by peer reviewing the PR. To streamline this process for the people reviewing these PRs, repositories make contribution guidelines. A pull request that for example, changes dozens of files and hundreds of lines of code without any description or context is extremely difficult to review, if it is reviewed at all and not immediately discarded. The aim of the contribution guidelines is to prevent this type of pull requests, although their specifics can vary across projects.

This streamlining process can be a bit of a double-edged sword, however. Since it is not always clear to newcomers how they should handle making these pull requests in accordance with repository-specific contribution guidelines, which can create a technical barrier that can impede their joining process [16]. And because anyone from any location worldwide can help contribute to OSS projects, it is often infeasible to personally assist people in onboarding a project.

In this study, we intend to answer the following research question:

**RQ.** *How useful is the use of a chatbot to onboard newcomers to open source projects with respect to making pull request?*

---

<sup>1</sup>[www.github.com](http://www.github.com)

We developed a chatbot, built using Rasa Open Source<sup>2</sup> and Botfront<sup>3</sup>, for the JabRef repository<sup>4</sup>. We then test the bot by having fourteen newcomers evaluate its perceived usefulness using the Technology Acceptance Model (TAM) [8]. Following that, we assess the results of the experiment further to gauge its effectiveness. It becomes evident that the chatbot’s perceived utility and ease of use hold significant promise, affirming its potential as a valuable tool for aiding newcomers in grasping the guidelines for making a pull request within open source software (OSS) projects.

In summary, this study delves into the challenge of newcomers comprehending the guidelines for making a pull request in OSS projects. By employing a chatbot as a potential solution, we have examined its utility and user-friendliness among newcomers. In the upcoming chapters, we will delve into the background and related work of onboarding processes and software engineering bots in Chapter 2 and Chapter 3, respectively. After which we will elaborate on our methodology for development of the chatbot and its evaluation in chapter 4. We will then present the quantitative as well as qualitative analysis of our results in chapter 5. We discuss the implications of our findings in chapter 6. Following that we will discuss some of the limitations of our study in chapter 7, and finally in chapter 8, draw comprehensive conclusions and outline some potential improvements.

---

<sup>2</sup>[www.rasa.com](http://www.rasa.com)

<sup>3</sup>[www.botfront.io](http://www.botfront.io)

<sup>4</sup>[www.github.com/JabRef/jabref](http://www.github.com/JabRef/jabref)

## Chapter 2

# Background

In this chapter, we lay the groundwork by first discussing the process of onboarding newcomers to open-source projects. We then delve into the realm of chatbots, exploring their evolving role and applications. Lastly, we delve into the essential tools necessary for chatbot development.

### 2.1 Onboarding to Open Source Projects

GitHub<sup>1</sup> is a powerful social coding platform that allows developers to collaborate on software projects. However, people onboarding OSS projects may encounter hurdles that can be difficult to overcome on their own. OSS onboarding refers to the process of introducing and integrating new contributors into the development community and guiding them through the initial steps of understanding the project's goals, technical requirements, and guidelines. New contributors may be unfamiliar with the project's codebase or development processes, making it difficult for them to make meaningful contributions. Furthermore, project maintainers may have limited to no time and resources to devote to onboarding new contributors, complicating the process even further since people who are onboarding may have difficulty finding answers to their questions [16].

To address these issues, many project maintainers have created onboarding processes and documentation to help new contributors get up to speed quickly. They may include tutorials, documentation, and guidelines for contributing code, an example of which can be seen in Figure 2.1. It's worth noting that project guidelines, which can usually be found in files like CON-

---

<sup>1</sup>[www.github.com](http://www.github.com)

TRIBUTING.md or as a subsection in the README.md file within open source repositories, serve as essential references for understanding and adhering to proper contribution practices. But even with these resources, new contributors may struggle to understand the project’s codebase and development processes and require ad hoc guidance. Since the documentation might not be exhaustive enough. One area that can be particularly challenging for new contributors is submitting pull requests [16]. Pull requests are a critical component of the collaborative software development process on GitHub, as they allow developers to propose changes to a project’s repository and collaborate with other contributors to review and refine those changes. However, pull requests can also be complex and require a significant amount of time and effort to create and review.

## Contributing

We encourage everyone to contribute to Python and that’s why we have put up this developer’s guide. If you still have questions after reviewing the material in this guide, then the [Core Python Mentorship](#) group is available to help guide new contributors through the process.

A number of individuals from the Python community have contributed to a series of excellent guides at [Open Source Guides](#).

Core developers and contributors alike will find the following guides useful:

- [How to Contribute to Open Source](#)
- [Building Welcoming Communities](#)

Guide for contributing to Python:

Contributors	Documentarians	Triagers	Core Developers
<a href="#">Setup and Building</a>	<a href="#">Helping with Documentation</a>	<a href="#">Issue Tracker</a>	<a href="#">Responsibilities</a>
<a href="#">Where to Get Help</a>	<a href="#">Getting Started</a>	<a href="#">Triageing an Issue</a>	<a href="#">Developer Log</a>
<a href="#">Lifecycle of a Pull Request</a>	<a href="#">Style Guide</a>	<a href="#">Helping Triage Issues</a>	<a href="#">Accepting Pull Requests</a>
<a href="#">Running and Writing Tests</a>	<a href="#">reStructuredText Primer</a>	<a href="#">Experts Index</a>	<a href="#">Development Cycle</a>
<a href="#">Fixing “easy” Issues (and Beyond)</a>	<a href="#">Translating</a>	<a href="#">GitHub Labels</a>	<a href="#">Motivations and Affiliations</a>
<a href="#">Following Python’s Development</a>		<a href="#">GitHub Issues for BPO Users</a>	<a href="#">Core Developers Office Hours</a>
<a href="#">Git Bootcamp and Cheat Sheet</a>		<a href="#">Triage Team</a>	<a href="#">Experts Index</a>
<a href="#">Development Cycle</a>			

We **recommend** that the documents in this guide be read as needed. You can stop where you feel comfortable and begin contributing immediately without reading and understanding these documents all at once. If you do choose to skip around within the documentation, be aware that it is written assuming preceding documentation has been read so you may find it necessary to backtrack to fill in missing concepts and terminology.

Figure 2.1: Example of contribution page from the cpython<sup>2</sup>GitHub Page.



Project maintainers may establish contribution guidelines for pull requests to address these issues. These guidelines may include coding standards, testing requirements, documentation, and review processes. For example, a project may require that all code submitted via pull requests adhere to a specific set of coding standards to ensure consistency and maintainability. The contribution guidelines may also state that all proposed changes should be thoroughly tested to ensure that they do not introduce new bugs or errors. By establishing contribution guidelines for pull requests, project maintainers ensure that new contributors can submit high-quality, well-documented code that meets the project’s standards and requirements. This helps streamline the review process and reduce the burden on project maintainers, while also improving the overall quality and reliability of the software being developed. All these guidelines however may be difficult to comprehend for people onboarding a public repository.

## 2.2 Chatbots

In recent years, chatbots have emerged as a promising technology with the potential to revolutionize various aspects by assisting people in a wide range of tasks and interactions [4, 7, 6]. A chatbot is a program designed to simulate human conversation and engage in real-time interactions with users through text-based or voice-based platforms. These intelligent bots are capable of understanding natural language and providing relevant responses, thus enabling automated communication.

The idea of chatbots dates back to 1950 when Alan Turing proposed “The Imitation Game” where he discussed the potential of computers being able to think and partake in conversations [19]. With ELIZA (1966) being one of first notable programs which made having a conversation with natural language possible [20]. However only recently, thanks to advancements in AI, machine learning, and natural language processing, chatbots started gaining traction. Integrating Large-scale Language Models (LLMs) like ChatGPT<sup>3</sup> into chatbots has become a rising trend in software engineering. ChatGPT, a prominent LLM, is gaining attention as a versatile code discussion bot, offering suggestions, explanations, and even generating code [18].

Now, these chatbots are not just chatterboxes. They are like mentors, much like the ones in education, where they guide learners step-by-step, recommend things, and give insights [21]. In the software development world, these chatbots could take the same role as a mentor to newcomers dealing

---

<sup>2</sup>[www.github.com/python/cpython](http://www.github.com/python/cpython)

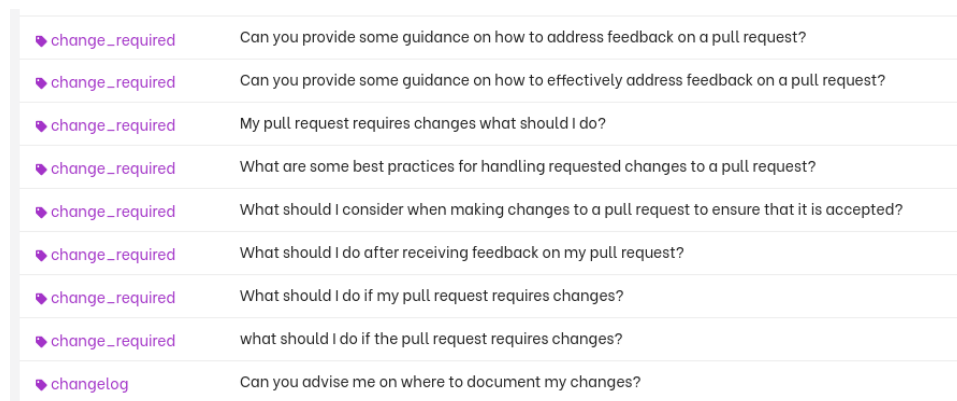
<sup>3</sup>[www.chat.openai.com](http://www.chat.openai.com)

with hurdles in open-source projects. Lessening the need for help from experienced developers, which can be hard to find in OSS development. It's important to note, however, that most bots in software development are currently limited to executing pre-defined scripts, lacking the dynamic interactions seen in education [10].

## 2.3 Chatbot Development

### 2.3.1 Rasa

Rasa Open Source<sup>4</sup> is an open-source conversational AI framework that enables the creation of chatbots. It provides a set of building blocks, such as Natural Language Understanding (NLU), stories, intentions, and rules. The NLU enables the chatbot to interpret prompts given by the user. You can then create intentions by defining the goals or objectives that the user wants to achieve through their message. For example, if a user asks, "How do I make a good commit message", the intention could be to get information about the requirements for a good commit message. To train the bot to map users' prompts to the right intention, it is trained on training data with example sentences of the different intentions.



change_required	Can you provide some guidance on how to address feedback on a pull request?
change_required	Can you provide some guidance on how to effectively address feedback on a pull request?
change_required	My pull request requires changes what should I do?
change_required	What are some best practices for handling requested changes to a pull request?
change_required	What should I consider when making changes to a pull request to ensure that it is accepted?
change_required	What should I do after receiving feedback on my pull request?
change_required	What should I do if my pull request requires changes?
change_required	what should I do if the pull request requires changes?
changelog	Can you advise me on where to document my changes?

Figure 2.2: NLU example, multiple training sentences that have the same intention.

---

<sup>4</sup>[www.rasa.com](http://www.rasa.com)

Stories are used to define conversation flows and describe how the chatbot should respond to different intentions. It is then possible to make different branches of stories to handle multiple conversational flows. Rules allow for specific behaviors for the chatbot, such as always responding with a greeting when prompted with a “greeting” intention, such as “Hello” [2].

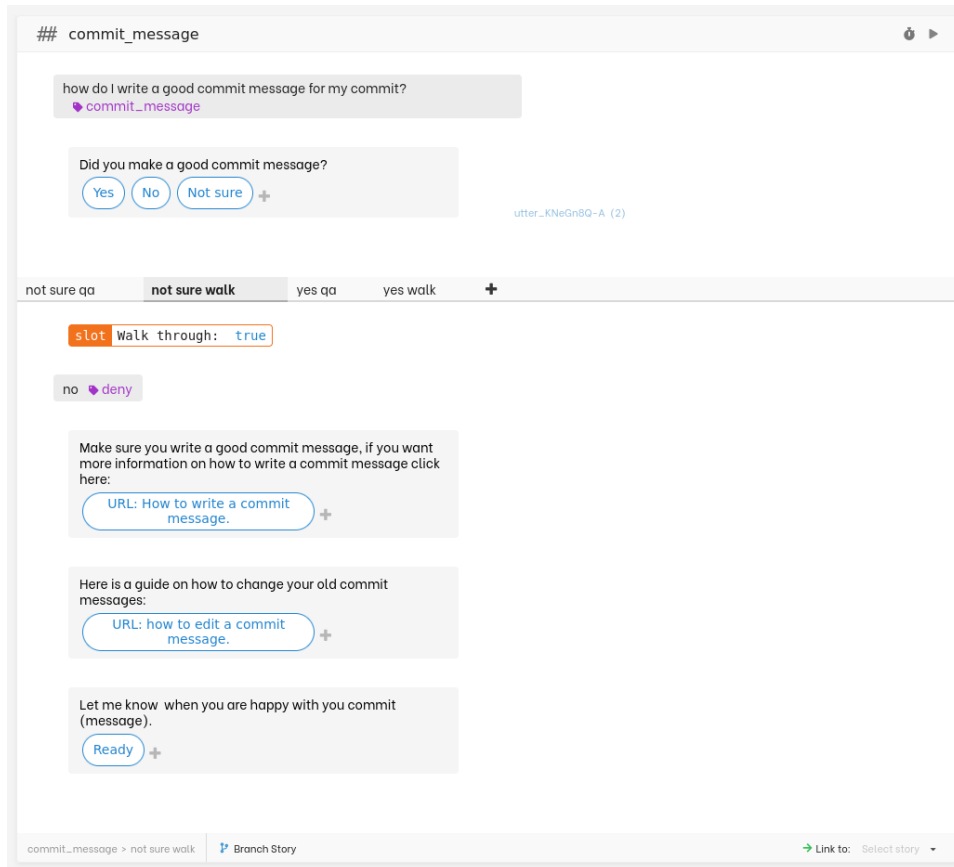


Figure 2.3: Story example

### 2.3.2 Botfront

Botfront<sup>5</sup> is an open-source platform that complements Rasa by offering a user-friendly interface for chatbot development and management. It provides a visual chatbot builder, making it easier to design conversational flows, define intents and manage stories [1]. Examples of the Botfront interface are presented in the figures 2.2 and 2.3.

---

<sup>5</sup>[www.botfront.io](http://www.botfront.io)

## Chapter 3

# Related Work

In this chapter, we delve into the existing research and developments that set the stage for our study. We begin by exploring the role of bots in software engineering, where they have become integral in various aspects, from development support to efficient team and task management. After which, we look into the the research done on the challenges of onboarding newcomers to open-source software projects.

### 3.1 Bots in Software Engineering

Bots have become commonplace in software engineering, offering invaluable assistance in various aspects including development and deployment support, as well as facilitating efficient team and task management. Notably, researchers like Lin et al. have explored how developers harness bots within platforms like Slack, showing that developers currently use bots for development and deployment support [12]. And Erlenhov et al. presented a mixed-method empirical study of DevBot usage in industrial practice, identifying distinct personas among software engineering bots users and highlighting the need for consensus and development of more advanced and transformative “smart” bots, bots that are unusually good or adaptive at executing task, in software engineering [10]. These papers help us better our understanding of the roles bots play within software engineering.

The goals of these bots is often to improve the productivity of developers. Storey et al. aimed to inspire researchers to study both the positive as well as the negative impact of bots on software development. So, they proposed a cognitive support framework for how bots can support software develop-

ment which aims to help with making bots more efficient and effective [17]. Understanding how bots influence software development is crucial for refining newcomer onboarding, aligning with our study’s focus on the usefulness and ease of use of such tools.

Traditionally these bots consisted mostly of plugins, scripts and architectures but conversational chatbots have started gaining popularity. Matthies et al. proposed a slack bot to support analyses and measurements of teams’ project data [14]. Another role that chatbots can take is that of a tutor. Hobert’s research delves into the potential role of chatbots in supporting novice programmers, particularly in instances where human teaching assistants or other forms of support are not available. His design science research project aims to create and evaluate a chatbot-based learning system that aids novices in learning software code. The study indicates that a chatbot is suited to take the role of a teaching assistant or lecturer [11]. Daud et al. and Chinedu et al. have showed that chatbots can provide significant support to novice programmers when learning Java and Python respectively [7, 6]. This aligns with our study’s exploration of chatbots’ capabilities in assisting newcomers in understanding the guideline for making pull requests within open source software projects.

In recent times, developers have begun to leverage Large Language Models (LLMs) for assistance in their tasks. Ross et al. have introduced an intriguing concept, suggesting that engaging in conversations with a code-fluent LLM, as opposed to simply invoking it, could potentially enable additional knowledge and capabilities beyond code generation [15]. This notion sheds light on potential future directions where chatbots could evolve to provide more comprehensive and interactive support for developers.

## 3.2 Onboarding to OSS projects

Efficiently onboarding newcomers is crucial for online communities that rely on the input of external participants. For this reason, Steinmacher et al. organised the barriers newcomers encounter when making their initial contributions to OSS projects [16]. To overcome some of these barriers Canfora et al. have suggested a system called Yoda (Young and newcOmer Developer Assistant) designed to identify and suggest mentors in software projects through the analysis of data extracted from mailing lists and versioning systems [5]. That way newcomers can receive help from mentors to overcome barriers they encounter. However, it is not always feasible to have a mentor help a newcomer. That is why Malheiros et al. introduced Mentor, a recommender system that assists newcomers in software projects by ana-

lyzing change requests and version control to provide relevant source code recommendations for solving these requests [13]. These studies highlight the critical importance of efficient onboarding processes and provide context for our exploration of using a chatbot to aid newcomers in understanding the guidelines for making pull requests.

Dominic et al. proposed a conversational bot that could help newcomers find new projects and assist them through the process of onboarding to the OSS community by providing helpful resources and recommending human mentors if needed [9]. Alves et al. implemented a chatbot to assist people in finding an appropriate task and compared the perceived usefulness of the chatbot to that of the GitHub issue tracker using TAM and found that mainly newcomers and inexperienced users perceive the chatbot as easier to use [4].

Building upon the work of Dominic et al. and that of Alves et al., whose chatbot was well-received by inexperienced users, our study tackles the challenge of newcomers encountering problems when making change request, a problem highlighted by Steinmacher et al. [16]. Through the implementation of a chatbot, we aim to offer real-time, tailored guidance, simplifying the onboarding process and enhancing newcomers' ability to navigate the specific contribution guidelines of open-source projects. By providing automated assistance, we also lessen the reliance on human mentors, making the onboarding experience more accessible.

## Chapter 4

# Methodology

In this chapter, we will begin by outlining the guidelines set for the case study repository which guide the development of our chatbot to provide information in alignment with these guidelines. Following this, we will delve into the implementation details, offering an in-depth understanding of how the bot was developed. Finally, we will discuss the structure of the questionnaire we used to evaluate the chatbot usefulness.

### 4.1 Chatbot Context

For this paper, we have chosen the JabRef repository<sup>1</sup> as our case study repository. JabRef is an open-source citation and reference management tool. It's important to emphasize that our focus on JabRef is for the purpose of a case study, and the insights gained from this study can, to some extent, be applied to other projects as well. We chose JabRef due to its existing and clear documentation for contributing, which can be found on their documentation website<sup>2</sup> which they link to in the contribution section of their README.md as seen in Figure 4.1. However, it's worth noting that all the guidelines are consolidated on a single extensive page, which could potentially pose challenges for newcomers who might unintentionally overlook certain sections. In the documentation for contributing, we identified the following basic guidelines:

---

<sup>1</sup>[github.com/JabRef/jabref](https://github.com/JabRef/jabref)

<sup>2</sup>[devdocs.jabref.org/contributing](https://devdocs.jabref.org/contributing)

1. Get the JabRef code on your local machine.
  - (a) Fork the JabRef into your GitHub account.
  - (b) Clone your forked repository on your local machine.
2. Create a branch for each improvement you implement.
3. Do your work on the new branch, not the main branch.
4. Create a pull request.
5. In case your pull request is not yet complete or not yet ready for review, consider creating a draft pull request instead.

And we have identified the following formal requirements for a pull request, aimed at both attributing credit to the contributor and maintaining clarity in the contribution process:

- Add your change to “CHANGELOG.md”
- Format of keyboard shortcuts
- Author credits: Do not add yourself at JavaDoc’s @authors
- Write a good commit message
- Test your code
- When adding a new “Localization.lang” entry, add new Localization.lang(“KEY”) to a Java file
- When adding a library, describe the library at external-libraries.md.
- When making an architectural decision, document your decision.

During the creation of our chatbot, we decided to exclude step 1b from the basic contribution guidelines in our implementation of the chatbot. This decision was based on several factors. Firstly, handling the variation that comes with different operating systems would have introduced unnecessary complexity to the chatbot. Additionally, the target audience of the chatbot mainly consists of programmers who are presumed to have some experience with Git(Hub). Finally step 1b does not impact the results of the pull request. Because of these reasons we decided its exclusion was acceptable.



## Contributing



Want to be part of a free and open-source project that tens of thousands of scientists use every day? Check out the ways you can contribute, below:

- Not a programmer? Help translating JabRef at [Crowdin](#) or learn how to help at [contribute.jabref.org](#)
- Quick overview on the architecture needed? Look at our [high-level documentation](#)
- For details on how to contribute, have a look at our [guidelines for contributing](#).
- You are welcome to contribute new features. To get your code included into JabRef, just [fork](#) the JabRef repository, make your changes, and create a [pull request](#).
- To work on existing JabRef issues, check out our [issue tracker](#). New to open source contributing? Look for issues with the "[good first issue](#)" label to get started.

We view pull requests as a collaborative process. Submit a pull request early to get feedback from the team on work in progress. We will discuss improvements with you and agree to merge them once the [developers](#) approve. Please also remember to discuss bigger changes early with the core developers to avoid a waste of time and work. Some fundamental design decisions can be found within our list of [Architectural Decision Records](#).

If you want a step-by-step walk-through on how to set-up your workspace, please check [this guideline](#).

To compile JabRef from source, you need a Java Development Kit 20 and `JAVA_HOME` pointing to this JDK. To run it, just execute `gradlew run`. When you want to develop, it is necessary to generate additional sources using `gradlew generateSource` and then generate the Eclipse `gradlew eclipse`. For IntelliJ IDEA, just import the project via a Gradle Import by pointing at the `build.gradle`.

`gradlew test` executes all tests. We use [GitHub Actions](#) for executing the tests after each commit. For developing, it is sufficient to locally only run the associated test for the classes you changed. Github will report any other failure.

Figure 4.1: Contribution section of the JabRef GitHub page.

## 4.2 Chatbot Implementation Details

The chatbot utilized in this research was developed using Rasa Open Source<sup>3</sup> and Botfront<sup>4</sup>. The source code for the chatbot can be accessed in our GitHub Repository: Chatbot-for-making-pull-request-JabRef<sup>5</sup>.

<sup>3</sup>[www.rasa.com](http://www.rasa.com)

<sup>4</sup>[www.botfront.io](http://www.botfront.io)

<sup>5</sup>[www.github.com/2maikel1/Chatbot-for-making-pull-request-JabRef](https://www.github.com/2maikel1/Chatbot-for-making-pull-request-JabRef)

The chatbot has the capability to address questions concerning each guideline and formal requirement for creating a pull request in JabRef. Additionally, it can provide guidance on some broader GitHub-related topics, such as the naming conventions for forks or branches. This functionality was achieved by assigning dedicated responses to individual the guidelines and formal requirements, which are activated based on the intentions of the users' questions. Further details on these requirements and guidelines can be found in section 4.1 - Chatbot Context.

Two distinct modes were implemented: the “walkthrough” mode and the “open questions” mode. At the start of the conversation the user is asked if they want to be walked through the process or if they have specific questions and depending on their response they will be put into the appropriate mode.

The walkthrough mode was designed to assist users with little to no experience in navigating the contribution guidelines of the JabRef repository<sup>6</sup>. It follows a structured conversational flow, a snippet of which can be seen in Figure 4.2. The full conversational flow can be found in the GitHub repository alongside the source code<sup>7</sup>. By following this conversational flow, it provides users with a guided step-by-step approach to ensure they cover all the necessary aspects of the contribution guidelines. Once the user confirms they completed a step, the chatbot automatically progresses to the next question/requirement, ensuring comprehensive coverage of all the requirements. To further guide and speed up the conversations, buttons with quick replies were implemented into the chatbot.

On the other hand, the open questions mode was created to provide quick access to specific information. Users can directly ask for the information they require without having to go through all the preceding steps. This mode offers a more flexible conversational flow, allowing users to ask direct questions and receive prompt responses. In the walkthrough mode, users also have the option to ask specific questions; however, the chatbot will subsequently proceed to the next step in the conversational flow.

By incorporating both the walkthrough and open questions modes, we aim to cater to individuals with varying levels of familiarity with the JabRef repository and its contribution guidelines.

---

<sup>6</sup><https://devdocs.jabref.org/contributing>

<sup>7</sup>[www.github.com/2maikel1/Chatbot-for-making-pull-request-JabRef](https://www.github.com/2maikel1/Chatbot-for-making-pull-request-JabRef)

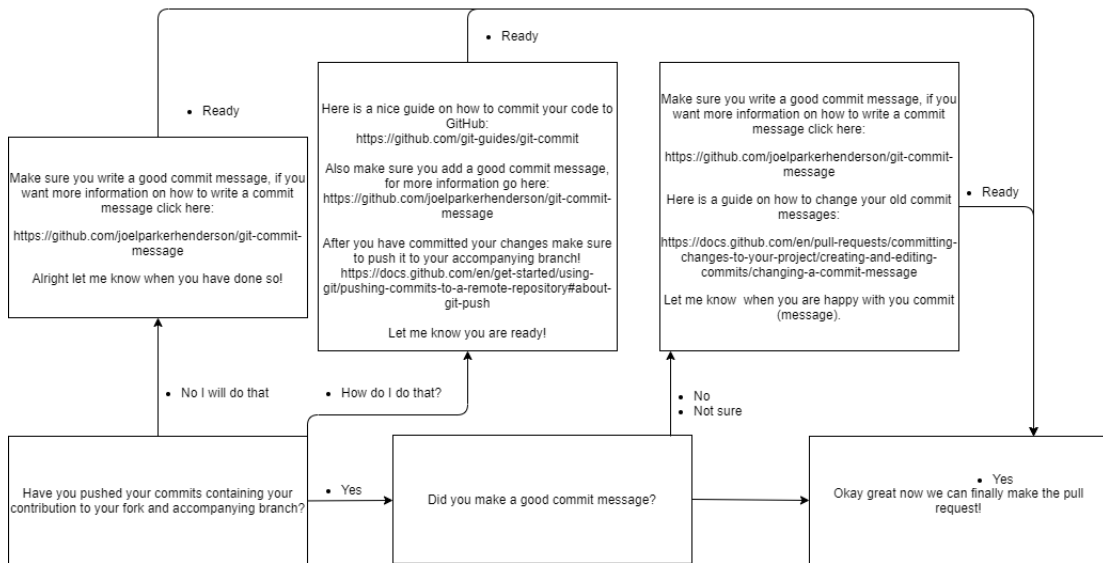


Figure 4.2: Illustration of conversational flow: How to push commits to your personal fork and branch.

#### 4.2.1 Website Hosting Chatbot

In order to provide users with access to the chatbot, we established a simple website. Visitors were instructed to open the chatbot interface positioned at the bottom right corner of the webpage. The site also displayed a warning advising users against sharing any sensitive information due to the absence of encryption in the communication process. Additionally, a button was implemented to enable users to reset or clear the chatbot conversation, this way you could have a fresh start if needed.

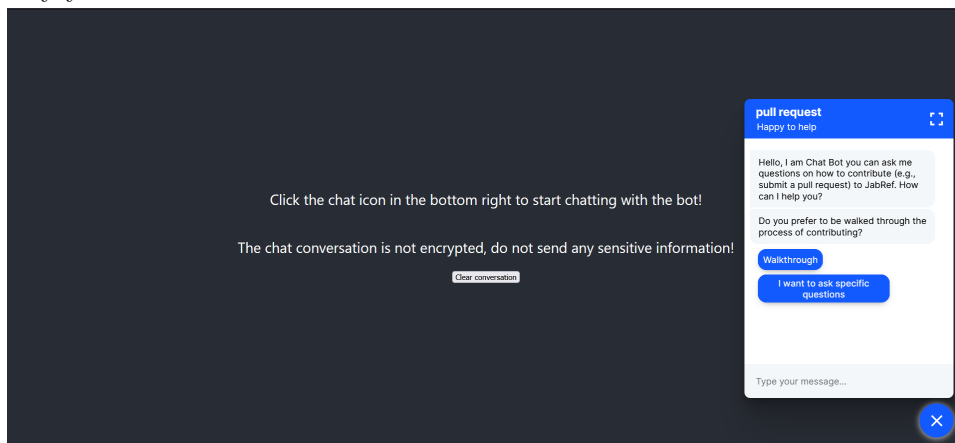


Figure 4.3: Screenshot of the webpage with the chatbot.

### 4.3 Questionnaire

To evaluate the usefulness our chatbot we used a structured questionnaire through a Google Forms survey, which consists of three distinct sections. For the first part we were interested to see if the perceived usefulness of the chatbot will be influenced by the level of experience. For this reason, we asked the following questions related to the participants' experience level:

- How much experience do you have with GitHub?
- How much experience do you have with contributing to open-source software projects?
- How much experience do you have with contributing to closed-source software projects?

Participants were provided with following multiple choice options to answer these questions:

- No Experience
- <1 year
- 1-2 years
- 3-5 years
- >5 years

Afterward, the participants were given the assignment of assuming the role of a contributor, considering that they were not actively contributing code. In this simulated scenario, participants were asked to complete the following task:

As part of this test, we will pretend that you have already implemented the changes requested. Therefore, your task is to identify what additional steps are necessary to create a pull request that meets the formal requirements of the repository.

For this exercise, we would like you to imagine that you have made the following changes to the repository:

- You have implemented a new feature that allows users to select all entries using the keyboard shortcut “ctrl + a”.
- You have fixed a bug related to the edit button by introducing a new library called “button”.

Your objective is to determine the next steps required to create pull request(s) such that it adheres to the formal requirements of pull request for the JabRef repository. Use the chatbot found here, to find out what these steps are. Note, the chatbot does not scale well to smaller/mobile screens, so if possible, use a computer.

Please do keep in mind that this chatbot is just a prototype so you might encounter some bugs. For this same reason we recommend using the provided buttons/quick replies, since those have been tested more thoroughly. If, however you want to ask something that is not one of the provided options feel free to ask the bot.

An optional text box to write down the required steps as well as a link to the chatbot were provided below this. The participants were told to move to the next part of the form once they thought they knew all the steps.

The second section of the form aimed to assess the chatbot’s effectiveness by asking whether participants found the necessary requirements for the given task. The participants were presented with a series of yes or no questions found in Table 4.1.

ID	Requirement Description
R1	You must create separate branches on your own fork for both changes you have implemented (i.e., the new feature and the bug fixing)
R2	You have to add your changes to the CHANGELOG.md
R3	You must format the keyboard shortcut like follows in the changelog: “<kbd>Ctrl</kbd> + <kbd>A</kbd>”
R4	You should not add yourself to JavaDoc’s @authors list
R5	You should write a descriptive commit message
R6	You must add newly added libraries to external-libraries.md

Table 4.1: Relevant formal requirements for the task.

The third and last part of the questionnaire was used to evaluate the participants’ experience with the chatbot based on the Technology Acceptance Model (TAM) [8]. TAM is a theoretical framework used to assess users’ acceptance and adoption of new technologies or systems. It consists of two primary factors: perceived usefulness (PU) and perceived ease of use (PEOU). In the context of the chatbot evaluation described, TAM was employed to measure participants’ experience and acceptance of the chatbot.

Perceived usefulness refers to the participants’ perception of how the chatbot enhances their ability to accomplish tasks effectively and efficiently. Perceived ease of use measures users’ perception of the ease and convenience of interacting with the chatbot. The questionnaire included three items to assess each of these factors which can be found in Table 4.2.

Factor	ID. Item Description
Perceived usefulness	PU1. Using the chatbot would enable me to accomplish tasks more quickly.
	PU2. I would find the chatbot useful when onboarding to an OSS project.
	PU3. The chatbot makes it easier to learn the formal requirements of an OSS project.
Perceived ease of use	PEOU1. I found the chatbot easy to use.
	PEOU2. The chatbot interaction was clear and understandable.
	PEOU3. Interacting with the chatbot was a flexible and adaptable experience for me.

Table 4.2: Questions adapted from the Technology Acceptance Model [8].

By analyzing participants’ responses to the TAM questions, we can gain insights into the perceived usefulness and ease of use of the chatbot. These findings can help evaluate the chatbot’s acceptance and inform potential improvements or modifications to enhance user experience. Additionally, an open-ended question was included to allow participants to leave additional feedback, providing qualitative insights into their overall experience with the chatbot.

Participants were also asked to indicate which mode(s) they used to interact with the chatbot, as explained in section 4.2. This question aimed to understand the participants’ preferences and choices regarding the interaction modes, providing insights into the impact of these modes on their experience.

# Chapter 5

## Results

In this chapter, we present the findings and results obtained from our study. We begin by assessing the usefulness of the chatbot. To achieve this, we perform quantitative analysis on the responses to the TAM questions provided by the participants. Additionally, we briefly examine the experience levels of the participants to gain insights into their backgrounds and potential impacts on the chatbot's effectiveness.

Following that, we perform a qualitative analysis on the open feedback given by the participants. This allows us to get a better understanding of their perceptions and experiences with the chatbot, providing valuable context and understanding beyond the quantitative data.

### 5.1 Quantitative Analysis

We received a total of 14 responses to the study including people following a computer science related study as well as people working for the Institute for Computing and Information Sciences of the university. The participants' experience levels were varied, with some having 3 to 5 years of active involvement in software development and using GitHub. Others had 1 to 2 years of experience, while a few had less than a year of experience. Some participants were entirely new to GitHub and open-source contributions. More detailed information about the experience levels of the participants that partook in the test can be found in Table 5.1. The results for the TAM related questions can be found in Figure 5.1.

Participant:	How much experience do you have with GitHub?	How much experience do you have with contributing to open-source software projects?	How much experience do you have with contributing to closed-source software?
P1	3-5 years	No experience	No experience
P2	1-2 years	No experience	>5 years
P3	>5 years	3-5 years	3-5 years
P4	3-5 years	3-5 years	3-5 years
P5	1-2 years	<1 year	<1 year
P6	1-2 years	<1 year	No experience
P7	3-5 years	No experience	<1 year
P8	1-2 years	No experience	<1 year
P9	1-2 years	<1 year	1-2 years
P10	3-5 years	<1 year	<1 year
P11	No experience	No experience	No experience
P12	1-2 years	No experience	1-2 years
P13	<1 year	No experience	No experience
P14	>5 years	3-5 years	3-5 years

Table 5.1: Experience levels of the participants

The TAM questions, found in Table 4.2, focused on participants’ perceptions of the chatbot’s usefulness and ease of use. The results of these TAM questions can be found in Figure 5.1. The responses show considerable variability, as indicated by the standard deviation values found in Table 5.2. While some participants rated the chatbot positively, others expressed lower levels of acceptance and ease of use. The median values pointed towards more positive perceptions among the participants. However, when considering the averages, the overall level of acceptance leaned slightly lower, though still positive. The individual answers for each participant can be found in Table A.1 in the appendix.

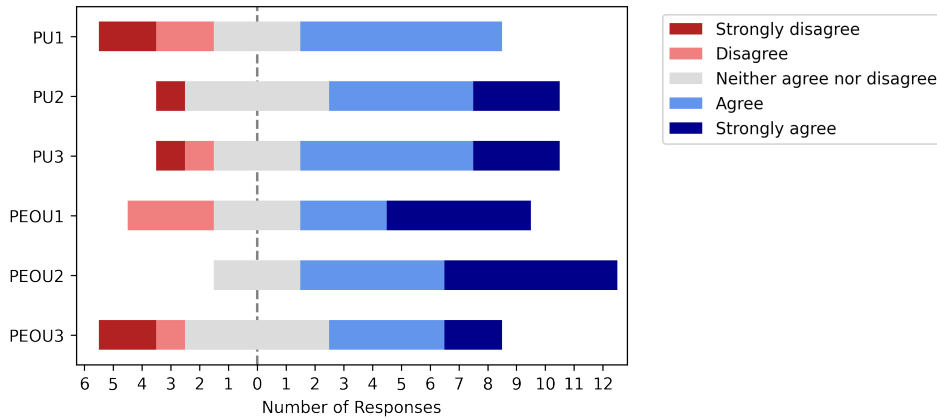


Figure 5.1: Responses to questionnaire

In Table 5.2 we can see that PU1: *Using the chatbot would enable me to accomplish tasks more quickly* and PEOU3: *Interacting with the chatbot was a flexible and adaptable experience for me.* scored lower compared to the



other PU and PEOU questions. This can suggest that the participants had relatively low perceptions of the chatbot’s ability to enhance task efficiency (PU1) and its overall flexibility and adaptability (PEOU3).

Question:	Average	Median	Standard Deviation
PU1	0.07	0.5	1.14
PU2	0.64	1	1.08
PU3	0.64	1	1.15
PEOU1	0.71	1	1.20
PEOU2	1.21	1	0.80
PEOU3	0.21	0	1.25

Table 5.2: Results of TAM questions, where “strongly disagree” through “strongly agree” are mapped from -2 to 2

Lastly, most participants found all the relevant requirements for the pull-request, as can be seen in Figure 5.2. There were however three participants that did not find requirement R3: *You must format the keyboard shortcut like follows in the changelog: “<kbd>Ctrl</kbd> + <kbd>A</kbd>”*. This might be explained by the fact that this requirement was only given if you respond “yes” to the following prompt: “Did you add keyboard shortcuts to the changelog?”. So, some of the participants might not have realised that they should have added a keyboard shortcut to the changelog at all, thus not being made aware of the formatting rules of the shortcut. Interestingly, all the participants that did not find this requirement had less than a year or no experience in both OSS and CSS. Which might explain their lack of knowledge about adding the shortcut to the changelog.

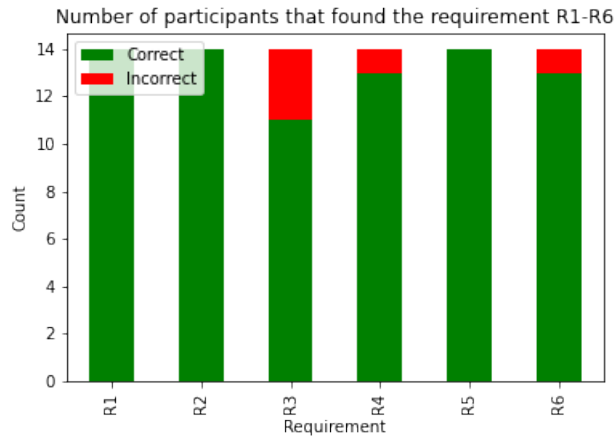


Figure 5.2: Number of correctly found requirements.

## 5.2 Qualitative Analysis of the Open-Questions

The results of the open-ended questions were grouped and themed. The feedback is presented in Table 5.3.

Categories		Occurrences
Communicability	Unwanted loopbacks	1
Maxim of Quantity	Cannot go into further depth	4
	Too much information on linked websites	1
	Takes too long	1
Profile Dependency	Lack of adaptability	3
Content	Verify if the user understands	1
	Lack of information about PR formatting	1

Table 5.3: Thematic analysis from participants open feedback.

**Communicability** encompasses feedback on the chatbot’s ability to understand and respond appropriately. One participant mentioned **unwanted loopbacks**, stating that *“It seemed to loop back to the walkthrough mode very quickly”* [P6]. This was a known issue and is discussed in section 6.1.1 - Chatbot Implementation Limitations.

**Maxim of Quantity** contains feedback regarding how informative the chatbot is without giving too much information. The most common point of feedback was about how the chatbot **cannot go into further depth**:

“I am not entirely sure that this is more useful than a regular webpage, as all the responses are rather formatted and formal still, so it seems very similar to a write-up online to me personally. If I e.g. ask the chatbot to go more in depth on a specific topic, or explain it in simpler terms, it just gives a response to another question, which is not ideal.” [P7]

Another participant also highlighted there is **too much information on linked website**:

“The way documentation is being offered could be improved a little bit. Some URL’s lead to the documentation pages containing information, guidelines and rules for several different steps and parts of the complete process. The general ‘Contributing — Developer Documentation’ page contains information about a lot

of different subjects, from commit guidelines, to specific JabRef file structures, to explaining how to update the changelog. The same big page is then provided at different interactions with the chatbot, while at that step only a small part of the document is actually needed. It would be clearer if only the exact information is needed for the current step in the whole process, also to prevent the wrong piece of documentation is used by accident. Also, sometimes the URL did not jump to the correct location in the page, which can be confusing and would be prevented by splitting the pages up. It could still be nice to have a overview of every different full documentation file, depending on the ones used in the walkthrough being followed, at the beginning or end of it.” [P12]

A different participant also expressed concern about the chatbot’s speed, stating that it **takes too long**:

“This process took so long that it would have been easier to read the information myself. The benefit of chatbots in my opinion is that they are a quick method to answer your very specific question. But this did not seem to do that.” [P6]

**Profile Dependency** contains feedback about how the chatbot adapts to the different needs of profiles/users. Another participant also highlighted the chatbot’s **lack of adaptability** to different user profiles and provided feedback regarding the limited walkthrough options:

“Currently it extensively describes both the global GitHub conventions, processes and documentation, as well as more JabRef specific rules, processes and documentation specific for the type of contribution. Being new to the JabRef project and making a first contribution to it, does not necessarily mean it is also the first time using GitHub. Seasoned programmers will probably know general naming conventions and processes by heart and only need JabRef specific help, like where specific files or documentation is located.

By adding multiple different walkthroughs by differentiating between the kind of contribution, developer skill and the degree of help needed.” [P12]

This same participant also suggested enhancing the chatbot’s effectiveness by incorporating various walkthrough modes. The participant expressed that the chatbot’s current extensive description of both global GitHub conventions and specific JabRef rules may not suit all contributors, especially those experienced with GitHub. They recommended implementing multiple walkthroughs tailored to different contribution types, developer skills, and help requirements.

“You could for instance offer the following three walkthroughs:

- In depth, step-by-step guide and documentation on the basics of the general GitHub process; such as branches, commits, pull requests, naming conventions and other standards
- A walkthrough describing only the process steps for JabRef specific practices, rules or contribution types, without asking if users need help with programming and GitHub basics.
- A walkthrough describing all steps, like the one that is being provided at the moment, with both global help, as well as specific help for different JabRef processes.

The new process could look something like this: after the type of contribution is chosen, multiple walkthroughs depending on the different needs of each user are offered, for instance the three described earlier: one with global GitHub help, one with JabRef specific help and one containing both. The specific walkthroughs can therefore skip questions, help and documentation depending on what the contributor needs, which means it can be completed faster. If the amount of information that has to be provided on each step is less, it will probably also be more clear to read and easier to structure. This will result in the overall chatbot interaction becoming more streamlined and user friendly.

Another use for different walkthroughs is that they could act as a refresher course or extra checklist for contributors returning to GitHub or JabRef after a (long) time so any possible changes to any conventions, rules or processes can be taken into account, or simply as additional support or verification for users when they feel the need for that.” [P12]

**Content** includes feedback on the core topics covered. One participant recommended the chatbot should **verify if the user understands** what was explained, which it currently does not do.

The last point of feedback was about the **lack of information about PR formatting**.

“None of the items raised by the chatbot are about how to format the comment in the PR so it really doesn’t add anything above the basics” [P3]

PR formatting is an important part of the contribution process, but since JabRef did not have any specific requirements stated, this was not included in our implementation of the chatbot.

## Chapter 6

# Discussion

This section provides additional insights on our results. Our quantitative results showed that the chatbot was mostly perceived as useful. Hence, we believe that a chatbot could serve as a useful tool for facilitating the onboarding process of individuals joining an open-source software project and seeking to understand the formal requirements for submitting pull requests. However, there are still some parts of the implementation that could be improved. As showed in table 5.2 the chatbot's ability to aide people in accomplishing tasks more quickly and the flexibility and adaptability of the chatbot are two of the main things that could be improved. These insights were reinforced by our qualitative analysis, with participants notably highlighting the need for more dynamic and varied responses.

One approach to achieve this is by using a combination of text summarization and natural language understanding techniques, applied to data mined from platforms like Stack Overflow or the project's existing documentation. This integration could potentially enable the chatbot to provide more flexible and relevant responses, aligning with the proposition of Dominic et al. [9]. Furthermore, a valuable suggestion put forth by participant P12 suggests the integration of multiple walkthrough modes, a feature that could enhance the chatbot's adaptability and reduce time waste by giving no/less unnecessary information.

Another area for potential improvement lies in enhancing the quality of the websites linked by the chatbot, as some of them appeared to be cluttered. In our present chatbot implementation, we utilized links to existing documentation to explain certain more extensive requirements. However, these existing pages occasionally contained an excess of information for the specific issue at hand. Developing streamlined pages tailored specifically for the chatbot could mitigate this issue of clutter and provide more concise and relevant guidance.

The sample size of our study is not large enough to draw statistically significant conclusions about the influence of experience levels on perceived usefulness. Despite this, we still observed interesting trends within the data. The data seems to indicate that there might be a positive correlation between the PU and the CSS experience, while also indicating that there might be a negative correlation between the PU and the OSS experience. This can be seen in Figure 6.1 and suggests that people that are more used CSS development perceive the extra guidance from the chatbot as more useful, while people with more OSS development experience might prefer more traditional resources.

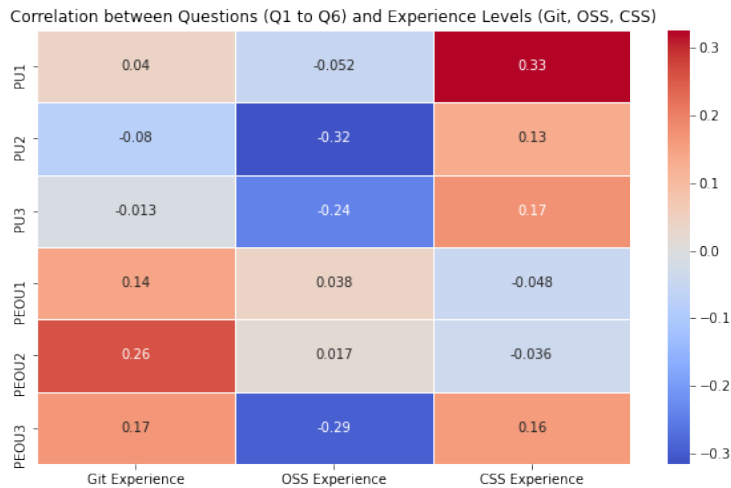


Figure 6.1: Correlation heatmap between experience and TAM questions

It is worth considering that individuals with experience in OSS development may be more accustomed to working independently, while those in CSS development might be accustomed to seeking and benefiting from additional guidance and support. This distinction in working styles could contribute to the observed trends in perceived usefulness.

## 6.1 Limitations

In this section, we will first delve into the limitations encountered during the implementation of the chatbot. After which we will discuss limitations encountered during the analysis of our results.

### 6.1.1 Chatbot Implementation Limitations

The current implementation of the chatbot has certain limitations that could be improved in future versions. For instance, the text on the buttons does not actually correspond to their intentions, especially when it comes to the walkthrough mode. So for example, a “no” button might be programmed with the intention of “How to make a branch”. This approach was chosen due to some difficulties encountered during development. When we programmed the stories to work with more generic intentions such as “deny” it would sometimes unexpectedly redirect the conversation to a different part of the story. So as a workaround, we programmed them to have more specific intentions, so they correctly go to the next part of the story. This however makes it so that the quick replies do not yield the same result as typing “no” into the chatbot manually. To limit the impact of this limitation users are encouraged to use buttons, if possible, provided that what they want to say aligns with one of the possible options.



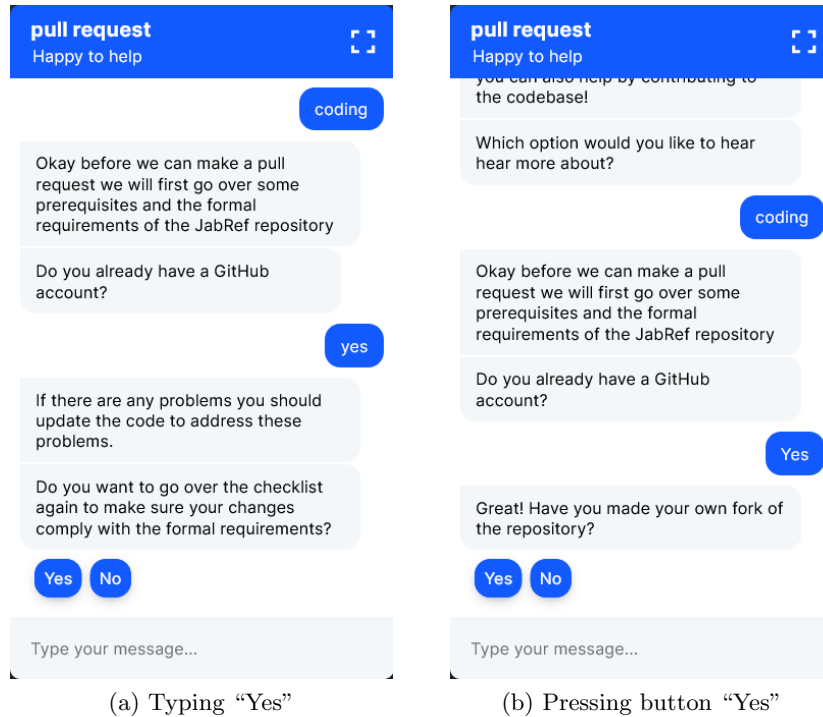


Figure 6.2: Example of how the text of the button does not always correspond to the intention. Here the button has an intention of “How do I make a fork?”

There are also a few more limitations of the walkthrough mode. Firstly, it lacks contextual awareness, making it unable to comprehend requests such as “can you provide further elaboration on this?” as it fails to identify the referent of “this.” Another limitation is that if you ask a question in walkthrough mode it will just “jump” to that part of the story. For example, if you were to go all the way through the walkthrough and then ask “What should I name my branch?” for example, it will then go over all the steps that come after the explanation of naming branches again, in order of the preprogrammed story.

### 6.1.2 Questionnaire Limitations

Due to the restricted number of participants, our ability to draw definitive conclusions about the influence of experience levels on the outcomes of the chatbot evaluation was somewhat constrained. While we aimed to include a varied group of participants with varying experience levels with GitHub and OSS contributions, the relatively small sample size of fourteen participants introduces limitations in terms of statistical significance. Despite this we observed some trends from the data collected. However, it's important to approach these findings with a degree of caution because of the small sample size.

Due to the limited amount of participant, we encountered an additional challenge in evaluating the chatbots performance. While our questionnaire provided valuable insights into participants' perceptions of the chatbot's perceived usefulness, we recognize that conducting a parallel evaluation of the usefulness of the existing documentation for the same repository would be beneficial. As this could have given us the opportunity to perform a comparative analysis of the two methods. However, because of the limited number of participants, we choose not to pursue this, as doing two separate evaluations could have further diluted the already small participant pool. Additionally, we opted not to conduct successive tests with the same participants, as knowledge gained from one test could influence their perception of the next tool used in the subsequent questionnaire.

## Chapter 7

# Conclusions

We proposed a new way to learn the requirements for pull request of OSS projects by letting users interact with a chatbot. To evaluate the usefulness and ease of use of this new bot we used the technology acceptance model. The results from this test shows that people do find the chatbot useful and easy to use.

Since the participants indicated that the chatbot was useful and easy to use, we believe that a chatbot can be a useful tool for guiding individuals new to open-source software projects in understanding the formal requirements of pull request.

There are, however, still ways the implementation of the chatbot could be improved. Notably, its ability to help accomplish task faster and adaptability could be refined. These findings, reinforced by participant feedback, underscore the need for more dynamic responses. By better addressing these areas, the chatbot could better aid newcomers with understanding the contribution guidelines for open-source software projects' pull requests.

# Bibliography

- [1] Botfront: powerful gui for rasa stories, rules and forms. Accessed on May 23, 2023.
- [2] Rasa: Open source conversational ai. Accessed on May 23, 2023.
- [3] Stack overflow developer survey 2022, 2022. Accessed on May 2, 2023.
- [4] Luiz Philipe Serrano Alves, Igor Scaliante Wiese, Ana Paula Chaves, and Igor Steinmacher. How to find my task? chatbot to assist newcomers in choosing tasks in oss projects. volume 13171 LNCS, pages 90–107. Springer Science and Business Media Deutschland GmbH, 2022.
- [5] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. Who is going to mentor newcomers in open source projects? In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE '12*, New York, NY, USA, 2012. Association for Computing Machinery.
- [6] Okonkwo Chinedu and Abejide Ade-Ibijola. Python-bot: A chatbot for teaching python programming. *Engineering Letters*, 29:25–34, 02 2021.
- [7] Siti Daud and Nurul Hidayah Mat Zain. e-java chatbot for learning programming language: A post-pandemic alternative virtual tutor. *International Journal of Emerging Trends in Engineering Research*, 8:3290–3298, 07 2020.
- [8] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13:319–340, 1989.
- [9] James Dominic, Jada Houser, Igor Steinmacher, Charles Ritter, and Paige Rodeghero. Conversational bot for newcomers onboarding to open source projects. pages 46–50. Association for Computing Machinery, Inc, 6 2020.

- [10] 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. pages 445–455. Association for Computing Machinery, Inc, 11 2020.
- [11] Sebastian Hobert. Say hello to ‘coding tutor’! design and evaluation of a chatbot-based learning system supporting students to learn to program. In *International Conference on Interaction Sciences*, 2019.
- [12] 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*, CSCW ’16 Companion, page 333–336, New York, NY, USA, 2016. Association for Computing Machinery.
- [13] Yuri Malheiros, Alan Moraes, Cleyton Trindade, and Silvio Meira. A source code recommender system to support newcomers. In *2012 IEEE 36th Annual Computer Software and Applications Conference*, pages 19–24, 2012.
- [14] Christoph Matthies, Franziska Dobrigkeit, and Guenter Hesse. An additional set of (automated) eyes: Chatbots for agile retrospectives. pages 34–37. Institute of Electrical and Electronics Engineers Inc., 5 2019.
- [15] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D. Weisz. The programmer’s assistant: Conversational interaction with a large language model for software development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI ’23, page 491–514, New York, NY, USA, 2023. Association for Computing Machinery.
- [16] Igor Steinmacher, Tayana Uchôa Conte, Marco Aurélio Gerosa, and David F. Redmiles. Social barriers faced by newcomers placing their first contribution in open source software projects. pages 1379–1392. Association for Computing Machinery, Inc, 2 2015.
- [17] 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, page 928–931, New York, NY, USA, 2016. Association for Computing Machinery.
- [18] Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawendé F. Bissyandé. Is chatgpt the ultimate programming assistant – how far is it?, 2023.

- [19] Alan M. Turing. Computing machinery and intelligence, 1950.
- [20] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 6:36–45, 1966.
- [21] Sebastian Wollny, Jan Schneider, Daniele Di Mitri, Joshua Weidlich, Marc Rittberger, and Hendrik Drachsler. Are we there yet? - a systematic literature review on chatbots in education. *Frontiers in Artificial Intelligence*, 4, 2021.

Appendix A

Appendix

Participants:	Participant 1:	Participant 2:	Participant 3:	Participant 4:	Participant 5:	Participant 6:	Participant 7:	Participant 8:	Participant 9:	Participant 10:	Participant 11:	Participant 12:	Participant 13:	Participant 14:
How much experience do you have with GitHub?	3-5 years	1-2 years	>5 years	3-5 years	1-2 years	1-2 years	3-5 years	1-2 years	1-2 years	3-5 years	No experience	1-2 years	<1 year	>5 years
How much experience do you have with contributing to open-source software projects?	No experience	No experience	3-5 years	3-5 years	<1 year	<1 year	No experience	No experience	<1 year	<1 year	No experience	No experience	No experience	3-5 years
How much experience do you have with contributing to closed-source software?	No experience	>5 years	3-5 years	3-5 years	<1 year	No experience	<1 year	<1 year	1-2 years	<1 year	No experience	1-2 years	No experience	3-5 years
You must create separate branches on your own fork for both changes you have implemented (i.e., the new feature and the bug fixing)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
You have to add your changes to the CHANGELOG.md	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
You must format the keyboard shortcut like follows in the changelog: "<kbd>Ctrl</kbd>+ <kbd>A</kbd>"	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
You should not add yourself to JavaDoc's @authors list	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
You should write a descriptive commit message	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
You must add newly added libraries to external-libraries.md	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
The chatbot has two different interaction modes: walkthrough and open questions. In the former, the chatbot walks you through the process of contributing to JabRef; in the latter, the bot allows you to ask specific questions. A mixture of both interaction approaches is also possible. Which one of the modes have you used?	I used the walkthrough mode	I used the walkthrough mode	I used the walkthrough mode	I used a mixture of both	I used the walkthrough mode	I used the open questions mode	I used a mixture of both	I used the walkthrough mode	I used the walkthrough mode	I used the walkthrough mode	I used a mixture of both	I used the walkthrough mode	I used a mixture of both	I used a mixture of both
How much do you agree with the following statement: Using the chatbot would enable me to accomplish tasks more quickly.*	4	4	1	4	4	1	3	2	4	3	3	4	2	4
How much do you agree with the following statement: I would find the chatbot useful when onboarding to an OSS project.	5	5	1	4	4	3	3	3	4	4	3	5	3	4
How much do you agree with the following statement: The chatbot makes it easier to learn the formal requirements of an OSS project.*	4	4	2	3	5	1	4	4	4	3	3	5	4	5
How much do you agree with the following statement: I found the chatbot easy to use.	5	2	3	5	5	2	3	4	5	4	2	5	4	3
How much do you agree with the following statement: The chatbot interaction was clear and understandable.	5	4	3	5	5	5	5	4	5	4	3	4	3	4
How much do you agree with the following statement: Interacting with the chatbot was a flexible and adaptable experience for me.	5	4	2	3	3	1	4	3	5	3	1	4	4	3

Table A.1: Raw results of the Google Forum with open feedback removed