The Power of Bots: Understanding Bots in OSS Projects

MAIRIELI WESSEL, University of São Paulo, Brazil
BRUNO MENDES DE SOUZA, Federal University of Technology, Paraná, Brazil
IGOR STEINMACHER, Northern Arizona University, USA and Federal University of Technology, Paraná, Brazil
IGOR S. WIESE, Federal University of Technology, Paraná, Brazil
IVANILTON POLATO, Federal University of Technology, Paraná, Brazil
ANA PAULA CHAVES, Federal University of Technology, Paraná, Brazil and Northern Arizona University, USA
MARCO A. GEROSA, Northern Arizona University, USA

Leveraging the pull request model of social coding platforms, Open Source Software (OSS) integrators review developers' contributions, checking aspects like license, code quality, and testability. Some projects use bots to automate predefined, sometimes repetitive tasks, thereby assisting integrators' and contributors' work. Our research investigates the usage and impact of such bots. We sampled 351 popular projects from GitHub and found that 93 (26%) use bots. We classified the bots, collected metrics from before and after bot adoption, and surveyed 228 developers and integrators. Our results indicate that bots perform numerous tasks. Although integrators reported that bots are useful for maintenance tasks, we did not find a consistent, statistically significant difference between before and after bot adoption across the analyzed projects in terms of number of comments, commits, changed files, and time to close pull requests. Our survey respondents deem the current bots as not smart enough and provided insights into the bots' relevance for specific tasks, challenges, and potential new features. We discuss some of the raised suggestions and challenges in light of the literature in order to help GitHub bot designers reuse and test ideas and technologies already investigated in other contexts.

CCS Concepts: • **Human-centered computing → Open source software**;

Additional Key Words and Phrases: Automated agents; pull request; bots; open source; pull-based model

ACM Reference Format:
Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. 2018. The Power of Bots: Understanding Bots in OSS Projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW, Article 182 (November 2018), 19 pages. https://doi.org/10.1145/3274451

1 INTRODUCTION

Open Source Software (OSS) developers have a long tradition of collaborating on platforms such as version control and bug reporting systems that support technical aspects of the development. Recently, new platforms that better support social interaction have been proposed: the so-called

Authors' addresses: Mairieli Wessel, University of São Paulo, São Paulo, SP, Brazil, mairieli@ime.usp.br; Bruno Mendes de Souza, Federal University of Technology, Paraná, Campo Mourão, PR, Brazil, brunosouza@alunos.utfpr.edu.br; Igor Steinmacher, Northern Arizona University, Flagstaff, AZ, USA , Federal University of Technology, Paraná, Campo Mourão, PR, Brazil, igor.steinmacher@nau.edu; Igor S. Wiese, Federal University of Technology, Paraná, Campo Mourão, PR, Brazil, igor@utfpr.edu.br; Ivanilton Polato, Federal University of Technology, Paraná, Campo Mourão, PR, Brazil, ipolato@utfpr.edu.br; Ana Paula Chaves, Federal University of Technology, Paraná, Campo Mourão, PR, Brazil , Northern Arizona University, Flagstaff, AZ, USA, anachaves@utfpr.edu.br; Marco A. Gerosa, Northern Arizona University, Flagstaff, AZ, USA, marco.gerosa@nau.edu.

"social-coding platforms" [12] (such as GitHub, GitLab, and Bitbucket), which are transforming collaboration in software development [22, 60, 61]. These platforms provide features that aid collaboration and sharing, such as pull requests, which have shaped software projects' work practices [14, 22, 44, 71]. Previous work has examined how the pull-based model affects OSS developers [23, 26, 42] and integrators' behavior [11, 24]. This model offers new opportunities for community engagement, but at the same time increases the workload for integrators to communicate, review code, deal with license issues, explain project guidelines, run tests, and merge pull requests [23].

To reduce the workload with repetitive tasks, OSS communities have been adopting bots, which are software agents that integrates their work with humans' tasks [16], serving as a conduit between users and services [55] and sometimes performing complex tasks that cannot be entirely automated [31]. Some bots ("chatbots" or "chatterbots") interact with humans through a conversational interface using natural language [31]. According to Farooq and Grudin [16], integration implies partnership, which means that the partners complement each others' activities. In this paper, we analyzed bots in OSS projects (more specifically on GitHub) that have a user profile and play a role within the development team, executing well-defined tasks that complements other developers' work.

Bots are extensively proposed and analyzed in the literature of different domains, including social media [1, 47, 70], online learning [20, 30, 38, 45], and Wikipedia [10, 19]. For collaborative software engineering, some preliminary studies seek to understand bots and how they are used to interact with messaging tools [34] and social media [41]. More specifically, some studies on bots focus on continuous integration builds and deployment [63]. Although some preliminary work focuses on the use of bots in software development projects [31, 55, 63], little is known about bots in OSS and the challenges they impose from the integrators' and contributors' perspectives. Therefore, in this paper we investigate how often popular software projects hosted on GitHub adopt bots. We also report a quantitative analysis of activity indicators before and after bot adoption and a qualitative analysis on how contributors and integrators perceive the relevance of bot support. We collected data from 351 OSS projects and surveyed 205 contributors and 23 integrators.

From our data analysis, we make the following contributions: (i) bring attention to bots, a relevant yet neglected resource that offers support for collaborative tasks in OSS; (ii) characterize the usage of bots in OSS projects; (iii) elucidate how contributors and integrators see the importance and support of bots; and (iv) present challenges introduced by bots and features that could enhance current support.

2 BACKGROUND

The origin of bots dates back to 1950, when Alan Turing proposed that machines could think [62]. Since then, the interaction between computers and humans has been a challenge for researchers [13, 65, 75]. Recently, advances in fields such as Artificial Intelligence, Natural Language Processing, and Machine Learning have enabled a proliferation of bots in several domains and the establishment of partnerships in which computers and humans construct meaning around each other's activities [16]. Bots enhance collaborative work [18] and influence changes in the workplace [33].

Technology enterprises have invested effort in developing bots as intelligent personal assistants, such as Apple's Siri [68] and Google Assistant [49], using conversational interfaces to help users perform a wide set of personal tasks. In contrast, thousands of bots perform single, particular tasks in a narrow domain of expertise [13]. For example, bots have been applied in the educational context [27], focusing on students' engagement [4, 6, 17], self-guided learning [40], course advising [28], tutoring [56, 57], and coaching [35]. There are also bots in marketing, e-commerce [36, 58], and customer services [21, 25]. Bots have also played a relevant role in peer production communities, such as Wikipedia [10, 18] and social networks [1].

In Software Engineering, bots support several activities, such as communication and decision-making [55]. Previous studies on how developers use bots in a popular messaging tool evidenced that bots support both technical and social activities [34]. In collaborative software development environments, bots automate tasks that generally require human interaction [31]. For example, Urli et al. [63] propose a bot that serves as a program repair tool-chain for continuous integration build failures in projects hosted on GitHub. Pérez-Soler et al. [41] developed a bot that interacts with developers via social networks to orchestrate collaborative software modeling. Beschastnikh et al. [5] proposed the use of bots as a solution for the automated deployment and evaluation of software engineering analysis techniques. However, while these studies provide recommendations on how to develop bots, as well as evaluate bots' capabilities and performance, they do not draw attention to the impact of bot adoption on software development or how software engineers perceive the bots' support.

Towards understanding the practical implications of bot adoption, Storey and Zagalsky [55] describe a cognitive framework to explain how bots support software development productivity. The framework identifies the roles bots play for different phases of the software development lifecycle, as well as the bots' contributions to developers' efficiency and effectiveness. Paikari and van der Hoek [39] introduce a framework to examine the current state of bots and identify directions for future work. Mirhosseini and Parnin [37] suggest that bots can encourage project maintainers to update dependencies and Lebeuf et al. [32] investigate how bots can potentially mitigate collaboration breakdowns. Although these papers analyzed the role of bots in software development, they do not focus on how contributors and integrators perceive the bots during their activities and the effects of bot adoption.

3 METHOD

This study aims to understand bot usage on GitHub projects, answering three research questions:

RQ1. How common are bots in GitHub projects?

We aimed to understand how commonly OSS projects use bots and what they use them for by manually analyzing a subset of the most starred projects from GitHub.

RQ2. How do the characteristics of pull requests compare before and after the bot adoption?

By answering this question, we want to understand whether the acceptance rate, interaction, and decision-making time of a project change after the bot adoption. We quantitatively analyzed pull requests of 44 software projects, investigating the number of merged and unmerged pull requests, number of commits, time-to-close, and number of comments.

RQ3. How do contributors and integrators perceive bots' support during the pull request submission process?

In this research question, we analyze the contributors' and integrators' perspectives by means of a survey aimed at understanding: (1) whether stakeholders perceive the presence of bots on pull request that they submit/merge; (2) whether stakeholders agree about the relevance of bot support on software tasks; (3) problems/challenges of using bots; and (4) missing features.

3.1 Selecting OSS Projects

We selected OSS projects hosted on GitHub (excluding non-software projects, such as textbooks or bookmarks) that received at least 2.5k stars before August 2017 (number of stars is a proxy for

popularity [7]). We started with 4,037 projects. To conduct our analysis, we sampled 351 projects, providing us a confidence level of 95% with a ±5% confidence interval.

To identify a bot, we verified whether it had a GitHub account and analyzed its name and description looking for bot references (e.g., *Bootstrap's Pull Request Checker* bot[1]). We also analyzed pull requests, looking for message patterns that could indicate that a profile is a bot (e.g., "This is an automated pull request to..."). Additionally, we found bots that were properly tagged in the pull request messages (e.g., *Welcome* bot).

In each selected project, we manually classified the type of bots. There have been some attempts to categorize the roles of bots in software engineering, like Storey and Zagalsky [55], who considered multiple interaction channels (IRC, Slack, and HipChat). Specifically for GitHub bots, Paikari and van der Hoek [39] categorized the bots according to the interaction characteristics, but did not analyze the roles that bots play in the development process. Since the purposes of the previous taxonomies are different from the current study's goal, we analyzed the task performed by the bots in the context of pull requests received by OSS projects. Two researchers independently conducted the manual classification, followed by consensus discussions. At the end of this process, we identified 93 projects (26.5%) making use of at least one bot.

These bots supported a total of 113,414 pull requests submitted by 3,371 different contributors and merged by 370 integrators. To enable comparison, we retained only those projects that had been active for at least six months before and six months after the bot adoption. Moreover, when analyzing the quantitative data from before and after adoption, we noticed that many projects had too few contributors during one of these periods. Thus, we also discarded projects with 7 or fewer contributors in these periods (first quartile). At the end of this process, our dataset comprised 44 projects, including active, popular, non-trivial, and diverse OSS projects.

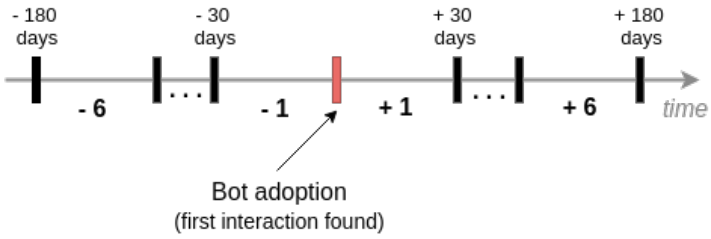


Fig. 1. Time Series overview

After identifying when a bot was adopted in the project, we collected data from 180 days before and 180 days after this event to conduct the quantitative analysis, as illustrated in Figure 1.

3.2 Understanding the developers' perspective

We conducted two surveys to understand how contributors and integrators perceive the adoption of bots to support pull requests. We analyzed pull requests submitted after the bot adoption and surveyed contributors who submitted pull requests and integrators who merged them.

The surveys were designed based on recommendations from Kitchenham et al. [29]. We also employed principles for increasing survey participation [48], such as sending personalized invitations, allowing participants to remain completely anonymous, and asking closed and direct questions. The contributor and integrator surveys were set up as online questionnaires.[2]

[1] https://github.com/mozilla/pdf.js/pull/8634

[2] The questionnaires can be found online at [https://github.com/mairieli/CSCW-2018/tree/master/questionnaires].

Both surveys were sent on December 18, 2017, and we received answers for a 30-day period. Participation was voluntary and the estimated time to complete each survey was 5-10 minutes. In our first survey, our target population comprised 3,371 contributors who made their valid e-mail addresses publicly available via the GitHub API. We received answers from 205 people. Our second survey was delivered to 368 integrators with valid e-mail addresses. We received 23 answers. For both surveys, we had a response rate of $\sim$ 6%. Our contributor survey had six questions:

Q1. Did you notice the participation of bots during your pull request submission/acceptance process? Choices: {yes, no}
Q2. Based on your experience, it was easier to contribute to the project... Choices: {When my pull request was assisted only by project members, When my pull request was assisted only by a bot; When my pull request was assisted by bots and members}
Q3. Based on your experience, rate the relevance (on a 5-point Likert Scale with neutral: Very Irrelevant, Irrelevant, Neutral, Relevant, Very Relevant) of bot support in your project for the following tasks: explain the project guidelines (e.g., explain how to contribute for new contributors); decrease code review effort; decrease time to merge/reject of pull request; automate the continuous integration task; run tests/quality assurance tasks; license issues related to contributions; improve the social interaction between integrators and contributors.
Q4. In your project, do bots support contributors/integrators in other tasks not mentioned above? If yes, in which tasks?
Q5. What are the problems/challenges of using bots in your project?
Q6. What features would you like to add to the bot(s) used in your project?

The integrators' survey comprised the last four questions (Q3-Q6).

3.3 Data analysis

First, we quantitatively collected and analyzed the number of projects using bots, the number of bots per project, and the types of bots used (**RQ1**). Second, to answer **RQ2**, we used the non-parametric Mann-Whitney-Wilcoxon (MWW) test [67] to identify whether there were differences among project indicators (*e.g.*, number of comments, number of commits, and close time) collected from pull requests before and after bot adoption. We also used Cliff's Delta statistic [46] to quantify the difference between these groups of observation beyond p-value interpretation. Moreover, we observed the monthly distribution of each metric to search for indications of changes.

In the third analysis, we used a card sorting approach to qualitatively analyze the answers to our surveys' open-ended questions (**RQ3**). Card sorting is a widely used technique for creating mental models and deriving taxonomies from data [**?**], evolving categories to identify common themes. Three researchers conducted this analysis 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 was followed by a discussion meeting until they reached consensus on the code names and on the 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. Our card sort was open, 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 (**RQ3**) to understand developers' perceptions about the relevance of bot support on software development tasks. In Section 4.3, we highlight the main themes that emerged along with quotes extracted from the open questions, which were chosen based on their representativeness.

4 RESULTS

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

4.1 RQ1. How common are bots in GitHub projects?

We identified 48 different bots in 93 projects. We classified the bots according to the tasks they perform:

Ensure License Agreement Signing (10 bots). Bots that comment on pull requests to direct contributors to sign a Contributor License Agreement (CLA) (e.g., Googlebot, Facebook Community bot, and meteor-bot).

Report Continuous Integration Failures (10 bots). Bots responsible for notifying contributors of test failures in CI tools (e.g., Elastic Machine, swift-ci, and The Travis Bot).

Review Code (7 bots). Bots that analyze source code (e.g., code style, code coverage, code quality, and smells) and give feedback (e.g., Hound bot – verifies code style violations; Coveralls bot – shows which parts of the source code are not covered by the test suite).

Review Pull Requests (7 bots). Bots that analyze pull requests and comment on them, identifying potential mistakes and how to fix them (e.g., Node.js GitHub Bot, Calypso Bot, and Bootstrap's Pull Request Checker Bot).

Assign Reviewers (7 bots). Bots that aks a maintainer to review the pull request (e.g., Rust highfive robot).

Welcome Newcomers (4 bots). Bots that send a welcome message to new contributors (e.g., Welcome bot).

Merge Pull Requests (4 bots). Bots that test pull requests and merge them if they pass (e.g., Kubernetes Submit Queue).

Scrap the Forge (3 bots). Bots external to the project that look for potential enhancements and automatically open pull requests. They look for typos in README files (ReadmeCritic), known vulnerabilities (Snyk bot), and outdated domain usage (npm-to-cdn-bot).

Run Automated Tests (2 bots). Bots that run tests to validate the changes made in the pull request (e.g., pdf.js test).

Build (2 bots). Bots designed to build the application upon request (Mary Poppins bot and pdf.js bot).

Control Dependencies (1 bot). Greenkeeper is a bot that informs when updates in the dependencies will break the software, reporting the analysis as a comment on the pull request.

Create Issues (1 bot). Doctrine Bot automatically opens issue on the bug tracker to help contributors.

Run Benchmarks (1 bot). Svelte-Bot is a bot that runs a benchmark and shows the results as a comment. It tests many application functions using different configurations – e.g., different versions and web browsers – to evaluate the application's performance.

The number of different bots per programming language is presented in Figure 2. To analyze how the projects' characteristics affect the use of bots, we show the domain, the main programming language, and the first bot adoption date for each project. We found 76 projects using only one bot, 16 projects using two bots, and only one project using three different bots. Ten of these bots performed two tasks per project, one bot performed three tasks, and the other 37 bots performed a single task. We noticed that bot adoption rose in popularity in 2014, with 17 adoptions. We also observed a similar rate of adoption by software projects in 2015 (19 projects) and 2017 (18 projects), with a peak in 2016 (31 projects). Regarding programming languages, JavaScript comprised the

largest sample of the selected projects. These projects normally use more than one bot. In contrast, we highlight Ruby, Objective-C, and Python, which represented more than 5 projects in our sample, but use a single bot.

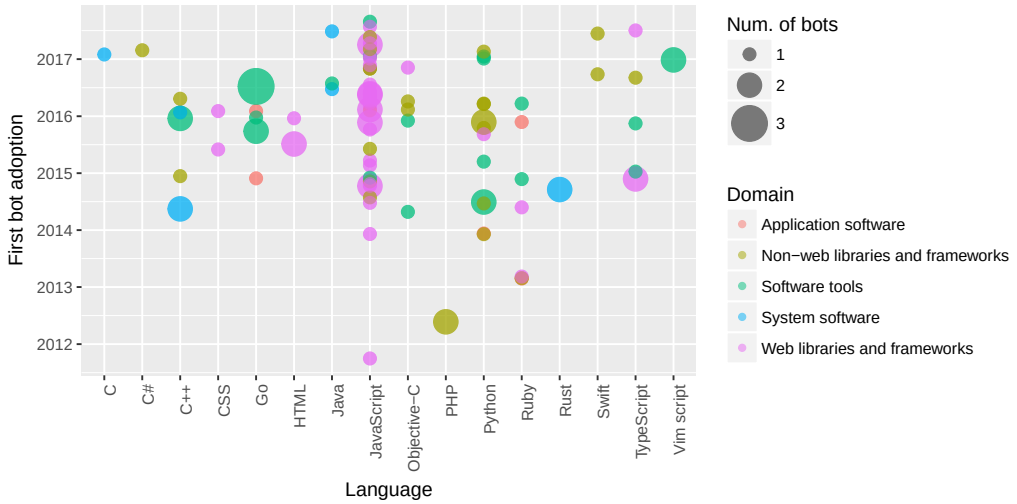


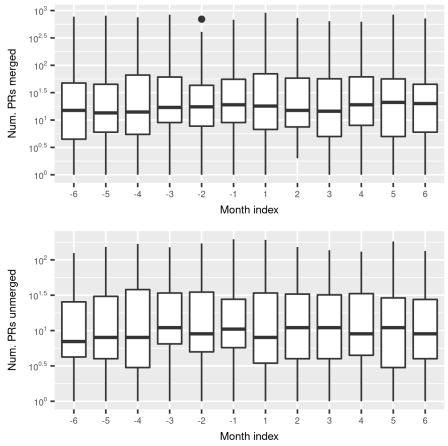
Fig. 2. Characteristic of projects *versus* bot adoption

RQ1. We found that almost one-third of the projects adopted at least one bot. The "boom" of adoption occurred after 2013. Projects commonly adopt bots to ensure license agreement signing, report continuous integration failures, assign reviewers, and automatically review the source code and the pull requests.

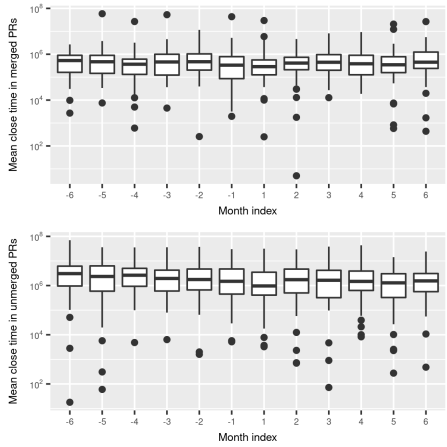
4.2 RQ2. How do the characteristics of pull requests compare before and after the bot adoption?

To look for changes in pull requests characteristics, we examined several metrics across the 44 selected projects, as can be seen in Figure 3. The top graphs refer to merged pull requests while the bottom graphs represent the unmerged pull requests for each six consecutive time intervals (months) before and after the first adoption of a bot. We used a log scale to normalize the distributions. The median values before and after bot adoption were, respectively: 15 and 17 for number of merged pull requests and 8 and 9 for unmerged; 131 and 116 hours for time to merge a pull request; 537 and 332 hours for time to reject a pull request; 19 and 27 for number commits per merged pull request and 16 and 19 for unmerged; and 2.7 and 3.1 comments for merged pull request and 3.9 and 3.7 for unmerged.

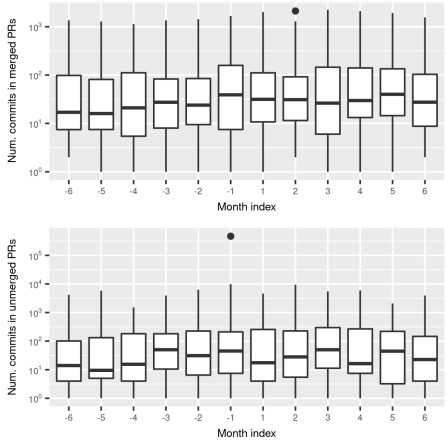
To verify whether we could evidence statistically significant differences for number of comments, commits, changed files, and close time for each project before and after bot implementation, we conducted Mann-Whitney-Wilcoxon tests for the different metrics. Our data shows that the percentage of the projects that showed statistically significant differences for merged and unmerged pull requests, respectively, were: (i) for the number of comments received per pull request, 66% and 39%; (ii) for number of commits per pull request, 34% and 25%; (iii) for number of changed files, 34% and 25%; and (iv) for time to close pull requests, 45% and 32%. The number of comments for merged



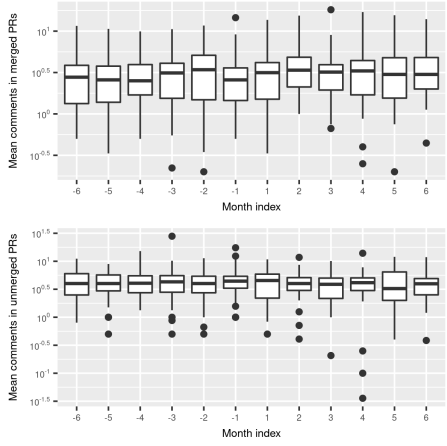
(a) Merged and unmerged pull requests



(b) Close time



(c) Number of Commits



(d) Number of Comments

Fig. 3. Distribution of metrics before and after bot adoption

pull requests presents the more noticeable difference, since we found a statistically significant difference for 29 out of 44 projects (66.0%). When analyzing the effect-size (using Cliff's delta), we noticed that the number of comments increased in 20 projects (5 with large, 4 with medium, 9 with small, and 2 with negligible effect-sizes), while it decreased for 9 (1 with large, 2 with medium, 4 with small, and 2 with negligible effect-sizes).

RQ2. The statistical results for each project showed that although there are statistically significant differences for some projects regarding the number of commits, number of changed files, and closed time before and after the bot adoption, the differences are not consistent across the projects. The most noticeable difference regarded the number of comments per pull request; however, for some projects the effect size indicated an increase and for others a decrease after bot adoption.

4.3 RQ3. How do contributors and integrators perceive the relevance of bot support during the pull request process?

When asked if they noticed the participation of bots during the pull request process, 200 (97.6%) contributors answered "yes." Moreover, as can be observed in Table 1, more than 90% of our respondents answered that they find it easier to submit a pull request when bots and project members collaborate during the review process.

Table 1. Contributors' answers on whether it is easier to contribute to the project when pull requests are assessed by bots or project members

Assisted by	# of answers (%)
Bots only	7 (3.4%)
Project members only	9 (4.4%)
Bots and project members	189 (92.2%)

We also asked contributors about their perception of the relevance of bots to support specific tasks. The answers followed a 5-point Likert scale with neutral, from "Very Irrelevant" to "Very Relevant." In Figure 4, we observe that most of the respondents perceived bots as helpful for most of the tasks. More than 90% of them highlighted that bots are relevant to automate continuous integration and run tests/quality assurance tasks. On the other hand, only 38% of the respondents somewhat agreed that bots are relevant to improve social interaction, which can indicate that they do not perceive bots as social proxies.

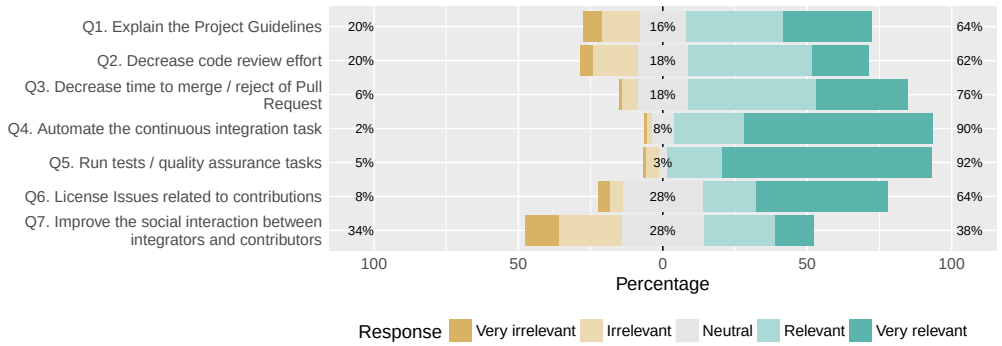


Fig. 4. Contributors' perceptions on the relevance of bot support in specific project tasks

Regarding integrators, we found that their perception was quite similar to the contributors, as shown in Figure 5. The main difference can be observed in Q1, for which 64% of the integrators did not agree that bots are capable of explaining project guidelines.

4.3.1 Challenges in using bots in pull requests. We openly asked contributors and integrators about the "problems/challenges of using bots." The challenges were grouped into 16 categories, as can be seen in Table 2.

From the developers' perspective, the most recurrent challenge with the current bots is that *bots have poor decision support mechanisms* (39 mentions). This challenge includes the cases in which the respondents say that bots "are not smart enough yet," which suggests they were expecting the bots to help them solve the hurdles, and not only raise problems, as a respondent explained: "bots

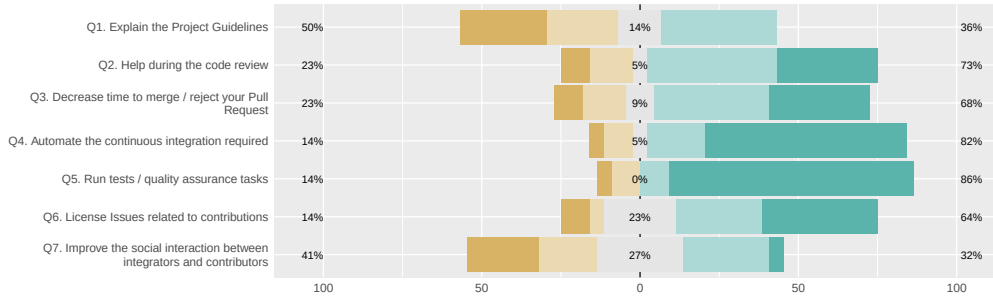


Fig. 5. Integrators' perceptions on the relevance of bot support in specific project tasks

Table 2. Problems/challenges introduced by bots in pull requests

Challenge	Contributors	Integrators
Bots have poor decision support mechanisms	39 (19.0%)	1 (4.5%)
Bots provide non-comprehensive/poor feedback	31 (15.1%)	–
Bots take wrong actions	18 (8.8%)	–
Bots are not able to review code	17 (8.3%)	–
Bots require manual intervention	9 (4.4%)	1 (4.5%)
Bots introduce communication noise	4 (1.9%)	–
Tailored bot configuration	3 (1.5%)	8 (36.3%)
Lack of information on how to interact with the bot	3 (1.5%)	1 (4.5%)
Bots have lack of guidance for newcomers	2 (1.0%)	–
Bots require high maintenance	1 (0.5%)	2 (9.1%)
Difficulty to coordinate multiple bots	1 (0.5%)	1 (4.5%)
Contributors don't understand the value of bots for maintainers	–	2 (9.1%)
Bots reduce human-human interaction	–	1 (4.5%)
Difficulty to find the appropriate bot	–	1 (4.5%)
Lack of available resources	–	1 (4.5%)
No challenge	79 (38.5%)	5 (22.7%)
Did not answer / N/A	23 (11.2%)	–

help streamline repetitive tasks, but cannot help with the decision-making, such as deciding if a build-passing/CI-approved request is indeed useful for the project at large." They also suggested that bots need to learn from previous interactions: "Sometimes they don't learn from other situations."

Several developers (31) also complained about the way the bots interact, saying that the *bots provide non-comprehensive/poor feedback*. In one example of bots' inability to provide adequate feedback, a respondent complained that bots can provide harsh responses, and thus "feel unfriendly in certain situations." This challenge relates to the way the bot is designed to provide answers, which can at times be misleading and unhelpful. One respondent mentioned that the bots do not provide "enough information on how to solve the issue," while another reported that "the worst problem is when the bot answers something that is not helpful."

With 18 mentions, *bots take the wrong action* was the third most reported problem of the current bots. In general, respondents complained that bots mistakenly close pull requests. One example was "closing an issue after X days of no activity (without having a single answer/post on the issue)." Other respondents mentioned race-conditions and test-flakes: "A significant challenge is coming across test-flakes…tests sometimes tend to fail for reasons completely unrelated to the changes being introduced through the patch."

Some other categories, although less recurrent, called our attention. Two of them refer to communication issues: *bots introduce communication noise*; and *lack of information on how to interact with the bot*. These challenges add to *bots provide non-comprehensive/poor feedback*, showing that it is necessary to improve the bot's usability. Another interesting challenge mentioned by respondents is that *bots lack guidance for newcomers*, which is a trending topic in OSS [53].

From the integrators' perspective, we observe that the challenges somewhat differ from those mentioned by contributors. It is clear that the challenges from the integrators' perspective relate more to choosing, setting up, and maintaining the bots. The most recurrent challenged mentioned by integrators refers to bots' configuration (*Tailored bot configuration*), which was reported by 8 respondents (36.3%), for instance: "configuring bots can be a lengthy process," and "designing the execution environment for our bot and giving it appropriate authority is difficult and takes time and effort." Two participants also mentioned that bots "are developed in an unstable way," so "they require high maintenance." We found other challenges: *difficulty to find the appropriate bot*; *difficulty to coordinate multiple bots*; and *lack of available resources*.

Notably, in addition to configuration and maintenance, two integrators reported that a challenge they see is that *"developers don't understand the value of bots for maintainers."* We could observe this fact from the feedback and the challenges reported on the developers' survey. One of them offered a nice explanation on that matter: "bots aren't magic, and sometimes it's hard for people to understand that they too are software that has to be carefully written. In other words, it's easy for people to put ungrounded faith or hope in a bot just because it's a bot."

4.3.2 Bot improvements. We also asked our participants about improvements and features they would like to add to bots. Most developers (55.6%) and 6 integrators (27.3%) did not suggest anything. The other answers were grouped, as can be seen in Table 3.

Table 3. Improvements proposed by developers and integrators

Features	Contributors	Integrators
Make the bots smarter	19 (9.3%)	–
Improve code review	16 (7.8%)	2 (9.1%)
Improve notification/awareness	11 (5.4%)	3 (18.2%)
Enhance user interaction	8 (3.9%)	2 (9.1%)
Recommend Related/similar PRs	5 (2.4%)	1 (4.5%)
Provide contribution guidelines	5 (2.4%)	1 (4.5%)
Improve feature communicability	5 (2.4%)	–
Recommend related artifacts	5 (2.4%)	–
Assign Reviewers	5 (2.4%)	–
Provide a way to request human help	4 (1.9%)	–
Release/deploy control	3 (1.5%)	2 (9.1%)
Provide guidelines for newcomers	2 (1.0%)	–
Keep the history of changes	2 (1.0%)	1 (4.5%)
Manage commit queues	2 (1.0%)	–
Estimate response time	2 (0.5%)	–
Calculate metrics	1 (0.5%)	1 (4.5%)
Answer specific questions	1 (0.5%)	–
Personal assistant	1 (0.5%)	–
Provide conversation statistics	1 (0.5%)	–
Label issues automatically	–	2 (9.1%)
Measure performance	–	1 (4.5%)
Offer polls to recommend new project features	–	1 (4.5%)
Resolve merge conflicts	–	1 (4.5%)

From the developers' perspective, the most recurrent suggestions relate to a desire to *make the bots smarter*, since "they are not smart enough yet." Most of the respondents mentioned that bots should learn from previous interactions and bring insights from the source code or issue tracker so they can provide better support to pull requests: "intelligence based on others issues and pull requests," and "bot could learn about all code on GitHub." One respondent, who previously mentioned a challenge related to flaky tests, proposed: "the ability for it to recognize test flakes by analyzing common failures often occurring across multiple pull requests over time, and to restart test jobs automatically when this happens." Still, some others provided more generic answers, briefly mentioning: "Machine learning" or "AI."

Another recurrent feature request from contributors regards *improving code review*. They mentioned different features, ranging from "check code style guidelines" to "auto-correct typical mistakes." More specifically, four developers suggested using "more linting code capabilities."

Improving the ways users can interact with bots was also mentioned. Since most bots are reactive and do not have any available interaction feature, 8 developers mentioned *enhance user interaction* as a good way to improve bots. One of them requested that: "it will be better to add more interactivity to some bots." *Improve notification/awareness* was also a recurrent category. Contributors mentioned better ways to remind project owners or reviewers about unresolved issues (e.g., "keep reminding repo owners about relevant issues that were not resolved or closed"), to send notifications about specific areas (e.g., "bots that remind me of any open issues in my areas of interest"), and notifications about the progress of the issue (e.g., "tracking all activity about this pull request"). Contributors also requested easier ways to learn how to interact with the bots, mentioning they need to know *how to communicate their features*.

In addition to the most recurrent ones, the developers mentioned as potential new features: *recommend related PRs* and *recommend related artifacts*. Additionally, two developers mentioned that bots would provide great value in receiving newcomers. One of them mentioned that a bot could be a "guide for first-time contributors. Where to get help from, e.g., mailing list, IRC, Slack." The second one mentioned that a bot could "guide people to low hanging fruits/issues and help them to engage in the project."

From the integrators' perspective, we noticed that improving interaction and receiving notifications from the bots are desired features. The most recurrent request refers to ways to *improve notifications/awareness*. However, as with the challenges, they focused on making easier integration and project maintenance. We can observe in Table 3 that integrators and developers mentioned features to provide *release/deploy control*, *keep the history of changes*, and *calculate metrics*, while only integrators mentioned the last four features on the table, which relate to improving project maintenance.

RQ3. Contributors and integrators reported 16 challenges for using bots in OSS projects. We found that 37% of these challenges were perceived by both contributors and integrators. Twenty four new features were also mentioned, with 33% of intersection in their suggestions. We observed that, while contributors want smarter bots that can recommend related artifacts and issues, and that have improved ways to interact, integrators reported both challenges and features related to facilitating bot and project maintenance.

5 DISCUSSION

An interesting result that emerged from the survey is the request for smarter bots. According to Farooq and Grudin [16], bots involve human-machine integration, which implies partnership. However, for OSS communities, this partnership still has room for improvement; as Table 2 shows,

several developers complained that *bots have poor decision support mechanisms* and *bots take wrong actions*. Moreover, developers and integrators provided an extensive list of desired features, such as *making the bots smarter*, *recommending related/similar PRs*, *assigning reviewers*, *answering specific questions*, and *automatically labeling issues*. Indeed, we could observe that the bots for GitHub OSS projects perform simple and repetitive tasks (Section 4.1), aiming to reduce integrators' effort. Little support is provided to the contributors at large. This poor support may justify the challenge mentioned by integrators that *developers don't understand the value of bots for maintainers*.

Existing literature charts many examples of bots that could help developers make decisions. Solutions to the aforementioned challenges could be inspired by intelligent interactive systems, which leverage techniques for rational decision coaching [35]. Bots could collaborate with developers by asking meaningful questions that lead them to decisions. Making smarter decisions (e.g., deciding the usefulness of a build) would require bots to be enriched with learning models for the target criteria (in the example, usefulness). This is a trending topic in other domains. For example, some bots in the education field learn from previous interactions and estimate students' interest level [38] or learning styles [30], adapting their interactions to improve collaboration. Similar models could be used in OSS development. Software Engineering literature on recommending systems explores, for example, issue/bug similarity techniques [2, 66, 74], approaches to retrieving related API or Q&A discussions by analyzing a piece of code or a question [43, 59, 73], and ways to recommend people given an issue or a pull request [3, 72]. To design smarter 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.

According to Lee et al. [33], "bots change the way people socialize when introduced to a community communication platform." However, in our survey almost 50% of the contributors and more than 50% of the integrators disagreed with the premise that bots improve social interaction. Moreover, from the answers to the open-ended questions, we observed that challenges and feature requests related to the way bots interact with developers, such as *bots provide non-comprehensive/poor feedback* and *bots introduce communication noise*. In addition, requests to *improve notification/awareness* and *enhance user interaction* can also be observed in the top requested features (Table 3). We learned from the literature that the customer service field had investigated similar problems. Gnewuch et al. [21] highlighted that most customer service bots could not meet users' interaction expectations. The authors listed issues such as not using appropriate language and giving generic information, and proposed requirements to develop enriched bots.

Better communicating a bot's features is another potential improvement on how bots interact with developers. According to Lebeuf et al. [31], this is important because "the bot's purpose — what it can and can't do — must be evident and match user expectations." HCI studies provide some steps toward understanding how bots should convey their features (what they can do, their context state, and intelligence level [25, 64]); from the CSCW perspective, it would be relevant to understand how the proposed approaches improve the success of accomplishing collaborative tasks.

Our study results also underscored that improving support for newcomer onboarding was mentioned both in challenges and as a desired feature. It is well-known that onboarding to an OSS community can be a journey full of obstacles [50]. This is especially relevant with the rise of casual contributors [42] and the number of quasi-contributors who do not succeed in contributing [52]. The literature offers potential ways to support newcomers and make this process smoother [9, 51, 54, 66, 69]. It would be beneficial for communities to take these approaches and guidelines into consideration to provide, as requested, *contribution guidelines*, or to offer an interactive way to help new members feel welcome and learn the contribution process. To provide this step, researchers

might find inspiration in the education domain, where bots have been proposed to advise students [20] and help students to share knowledge within communities [45].

From the integrators' perspective, we observed that the concerns and ideas about desired features pertain to bots' maintenance and setup. Bot developers need to consider the diversity of languages, frameworks, and platforms used in OSS projects. We believe that the setup and maintenance challenge can be explained by the maturity level of bots in the GitHub pull request context.

As decribed by Storey and Zagalsky [55], bots implement *"a conduit or an interface between users and services, typically through a conversational user interface."* Bots provide new forms of interactions with already existing tools [8], automating tasks and binding services together. However, GitHub bots are not as evolved as in other domains (e.g., education and customer service) and developers feel that bots should provide smarter ways to access current tools and services.

6 THREATS TO VALIDITY

While our results only apply to popular OSS projects hosted on GitHub, many relevant projects are currently hosted on this platform [15]. Our results are also limited by our selection of projects, which one might argue is small or non-representative. However, the selected projects are diverse in several dimensions, such as `domain`, `programming languages`, `popularity`, and `activity`. We also used a statistical approach to randomly select a sample of projects (see Section 3.1 for more details). Moreover, we likely did not discover all possible pull request characteristics that bots can influence. Even analyzing several projects, we could not find any statistically significant differences. This could be due to the presence of different bots with different goals and bots external to the projects, which can result in different outcomes. Future work can analyze specific types of bots. Moreover, most bots are designed to support maintainers facilitating their tasks. To complement our quantitative analysis, we conducted a survey to gain information from the stakeholders who interact with the bots. With our methodology and infrastructure, similar analysis can be conducted in the future to explore additional details about bot usage on GitHub. For replication purposes, we made our data and source code publicly available.[3]

Finally, since we leverage qualitative research methods to categorize the open-ended questions asked in our surveys, as well as to classify bots, we may have introduced categorization bias. To mitigate this bias, we conducted this process in pairs and carefully discussed categorization among the authors. Also regarding our surveys, the order that we presented the questions to the respondents may have influenced the way they answered them. We tried to order the questions based on the natural sequence of actions to help respondents understand the questions' context. We designed our survey to be short, with anonymous responses and voluntary participation.

7 CONCLUSION

In this paper, we showed that bots are rather common (one-third of the projects in our sample adopt them) and are used for a variety of tasks, such as ensuring license agreement signing, reporting continuous integration failures, reviewing code and pull requests, and assigning reviewers **(RQ1)**. When we quantitatively compared pull requests characteristics before and after bot adoption (**RQ2**), we could not find consistent statistically significant results across the analyzed projects. The only noticeable difference regards the number of comments per pull request, which increased for some projects and decreased for others. In our surveys (**RQ3**), most contributors and integrators see relevance of bot support for explaining project guidelines, decreasing code review effort and time to merge or reject pull requests, automate continuous integration tasks, and deal with license issues

[3] https://github.com/mairieli/CSCW-2018

for contributors. However, they manifested diverging opinions regarding the relevance of bots for improving social interaction in the project.

Contributors and integrators reported 16 challenges for using bots in OSS projects and 24 new features or improvements. In general, little support is provided for contributors' work and newcomers. While contributors want smarter bots, integrators want tailored bot configuration. As we discuss in the paper, the literature of the education, consumer service, software engineering, and HCI fields can help GitHub bot designers to enhance existing bots to achieve a higher level of partnership with contributors and integrators.

ACKNOWLEDGEMENTS

We would like to wholeheartedly thank each and every the developers who participated in our research. We also thank the reviewers for their valuable comments that made this paper stronger. This work is supported by the CNPq (Grant #430642/2016-4); FAPESP (Grant #2015/24527-3) and Northern Arizona University.

REFERENCES

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

[2] Karan Aggarwal, Finbarr Timbers, Tanner Rutgers, Abram Hindle, Eleni Stroulia, and Russell Greiner. 2016. Detecting duplicate bug reports with software engineering domain knowledge. *Journal of Software: Evolution and Process* 29, 3 (2016), e1821. https://doi.org/10.1002/smr.1821 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.1821

[3] John Anvik, Lyndon Hiew, and Gail C. Murphy. 2006. Who Should Fix This Bug?. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. ACM, New York, NY, USA, 361–370. https://doi.org/10.1145/1134285.1134336

[4] Luciana Benotti, María Cecilia Martínez, and Fernando Schapachnik. 2014. Engaging High School Students Using Chatbots. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14)*. ACM, New York, NY, USA, 63–68. https://doi.org/10.1145/2591708.2591728

[5] Ivan Beschastnikh, Mircea F. Lungu, and Yanyan Zhuang. 2017. Accelerating Software Engineering Research Adoption with Analysis Bots. In *Proceedings of the 39th International Conference on Software Engineering: New Ideas and Emerging Results Track (ICSE-NIER '17)*. IEEE Press, Piscataway, NJ, USA, 35–38. https://doi.org/10.1109/ICSE-NIER.2017.17

[6] Patrick Bii. 2013. Chatbot technology: A possible means of unlocking student potential to learn how to learn. *Educational Research* 4, 2 (2013), 218–221.

[7] Hudson Borges, Andre Hora, and Marco Tulio Valente. 2016. Understanding the Factors That Impact the Popularity of GitHub Repositories. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Washington, DC, USA, 334–344. https://doi.org/10.1109/ICSME.2016.31

[8] Nick C. Bradley, Thomas Fritz, and Reid Holmes. 2018. Context-aware Conversational Developer Assistants. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*. ACM, New York, NY, USA, 993–1003. https://doi.org/10.1145/3180155.3180238

[9] Gerardo Canfora, Massimiliano di Penta, Rocco Oliveto, and Sebastiano Panichella. 2012. Who is Going to Mentor Newcomers in Open Source Projects?. In *ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE '12)*. ACM, New York, NY, USA, Article 44, 11 pages. https://doi.org/10.1145/2393596.2393647

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

[11] Catarina Costa, Jair Figueiredo, Leonardo Murta, and Anita Sarma. 2016. TIPMerge: Recommending Experts for Integrating Changes Across Branches. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. ACM, New York, NY, USA, 523–534. https://doi.org/10.1145/2950290.2950339

[12] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *CSCW*. ACM, New York, NY, USA, 1277–1286.

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

[14] Manoel Limeira de Lima Júnior, Daricélio Moreira Soares, Alexandre Plastino, and Leonardo Murta. 2018. Automatic Assignment of Integrators to Pull Requests:The Importance of Selecting Appropriate Attributes. *Journal of Systems*

and Software 144 (2018), 181 – 196. https://doi.org/10.1016/j.jss.2018.05.065

[15] Luiz Felipe Dias, Igor Steinmacher, Gustavo Pinto, Daniel Alencar da Costa, and Marco Aurélio Gerosa. 2016. How Does the Shift to GitHub Impact Project Collaboration?. In *2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016, Raleigh, NC, USA, October 2-7, 2016*. IEEE Computer Society, Los Alamitos, California, USA, 473–477.

[16] Umer Farooq and Jonathan Grudin. 2016. Human-computer Integration. *interactions* 23, 6 (Oct. 2016), 26–32. https://doi.org/10.1145/3001896

[17] Luke K Fryer, Mary Ainley, Andrew Thompson, Aaron Gibson, and Zelinda Sherlock. 2017. Stimulating and sustaining interest in a language course: An experimental comparison of Chatbot and Human task partners. *Computers in Human Behavior* 75 (2017), 461 – 468. https://doi.org/10.1016/j.chb.2017.05.045

[18] R. Stuart Geiger. 2013. Are Computers Merely "Supporting" Cooperative Work: Towards an Ethnography of Bot Development. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work Companion (CSCW '13)*. ACM, New York, NY, USA, 51–56. https://doi.org/10.1145/2441955.2441970

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

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

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

[22] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 345–355. https://doi.org/10.1145/2568225.2568260

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

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

[25] Mohit Jain, Ramachandra Kota, Pratyush Kumar, and Shwetak N. Patel. 2018. Convey: Exploring the Use of a Context View for Chatbots. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 468, 6 pages. https://doi.org/10.1145/3173574.3174042

[26] David Kavaler, Sasha Sirovica, Vincent Hellendoorn, Raul Aranovich, and Vladimir Filkov. 2017. Perceived Language Complexity in GitHub Issue Discussions and Their Effect on Issue Resolution. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017)*. IEEE Press, Piscataway, NJ, USA, 72–83. http://dl.acm.org/citation.cfm?id=3155562.3155576

[27] Alice Kerry, Richard Ellis, and Susan Bull. 2009. Conversational agents in E-Learning. In *Applications and Innovations in Intelligent Systems XVI*. Springer, London, UK, 169–182.

[28] Hyekyung Kim, Miguel E Ruiz, and Lorna Peterson. 2007. Usability and effectiveness evaluation of a course-advising chat bot. *Proceedings of the American Society for Information Science and Technology* 44, 1 (2007), 1–5.

[29] Barbara A. Kitchenham, Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. 2002. Preliminary guidelines for empirical research in software engineering. *Software Engineering, IEEE Transactions on* 28, 8 (Aug 2002), 721–734. https://doi.org/10.1109/TSE.2002.1027796

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

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

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

[33] Minha Lee, Lily Frank, Femke Beute, Yvonne de Kort, and Wijnand IJsselsteijn. 2017. Bots mind the social-technical gap. In *Proceedings of 15th European Conference on Computer-Supported Cooperative Work-Exploratory Papers*. EUSSET), 35–54. https://hdl.handle.net/20.500.12015/2929

[34] Bin Lin, Alexey Zagalsky, Margaret;Anne Storey, and Alexander Serebrenik. 2016. Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW '16 Companion)*. ACM, New York, NY, USA, 333–336.

https://doi.org/10.1145/2818052.2869117
[35] Daniel Mäurer and Karsten Weihe. 2015. Benjamin Franklin's decision method is acceptable and helpful with a conversational agent. In *Intelligent Interactive Multimedia Systems and Services*. Springer, Cham, Switzerland, 109–120.
[36] Mohammed Slim Ben Mimoun, Ingrid Poncin, and Marion Garnier. 2017. Animated conversational agents and e-consumer productivity: The roles of agents and individual characteristics. *Information & Management* 54, 5 (2017), 545–559.
[37] Samim Mirhosseini and Chris Parnin. 2017. Can Automated Pull Requests Encourage Software Developers to Upgrade Out-of-date Dependencies?. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017)*. IEEE Press, Piscataway, NJ, USA, 84–94. http://dl.acm.org/citation.cfm?id=3155562.3155577
[38] Kazuaki Nakamura, Koh Kakusho, Tetsuo Shoji, and Michihiko Minoh. 2012. Investigation of a Method to Estimate Learners' Interest Level for Agent-based Conversational e-Learning. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, Berlin, Heidelberg, 425–433.
[39] Elahe Paikari and André van der Hoek. 2018. A Framework for Understanding Chatbots and Their Future. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '18)*. ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/3195836.3195859
[40] Juanan Pereira. 2016. Leveraging Chatbots to Improve Self-guided Learning Through Conversational Quizzes. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM '16)*. ACM, New York, NY, USA, 911–918. https://doi.org/10.1145/3012430.3012625
[41] Sara Pérez-Soler, Esther Guerra, Juan de Lara, and Francisco Jurado. 2017. 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)*. IEEE Press, Piscataway, NJ, USA, 723–728. http://dl.acm.org/citation.cfm?id=3155562.3155652
[42] Gustavo Pinto, Igor Steinmacher, and Marco A. Gerosa. 2016. More Common Than You Think: An In-depth Study of Casual Contributors. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. IEEE Computer Society, Los Alamitos, CA, USA, 112–123. https://doi.org/10.1109/SANER.2016.68
[43] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack Overflow in the IDE. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, Piscataway, NJ, USA, 1295–1298. http://dl.acm.org/citation.cfm?id=2486788.2486988
[44] Mohammad Masudur Rahman, Chanchal K. Roy, and Jason A. Collins. 2016. CoRReCT: Code Reviewer Recommendation in GitHub Based on Cross-project and Technology Experience. In *38th International Conference on Software Engineering Companion (ICSE '16)*. ACM, New York, NY, USA, 222–231. https://doi.org/10.1145/2889160.2889244
[45] Claudia Roda, Albert Angehrn, Thierry Nabeth, and Liana Razmerita. 2003. Using conversational agents to support the adoption of knowledge sharing practices. *Interacting with Computers* 15, 1 (2003), 57–89.
[46] Jeanine Romano, Jeffrey D. Kromrey, Jesse Coraggio, and Jeff Skowronek. 2006. Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen's d for evaluating group differences on the NSSE and other surveys?. In *Annual Meeting of the Florida Association of Institutional Research*. 1–3.
[47] Saiph Savage, Andres Monroy-Hernandez, and Tobias Höllerer. 2016. Botivist: Calling volunteers to action using online bots. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, New York, NY, USA, 813–822.
[48] Edward Smith, Robert Loftin, Emerson Murphy-Hill, Christian Bird, and Thomas Zimmermann. 2013. Improving developer participation rates in surveys. In *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE Computer Science, Los Alamitos, CA, USA, 89–92.
[49] Nick Statt. 2016. Why Google's fancy new AI assistant is just called 'Google'. Retrieved March 21, 2017 from https://www.theverge.com/2016/5/20/11721278/google-ai-assistant-name-vs-alexa-siri. The Verge. Archived from the original on March 21, 2017.
[50] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David F. Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 1–13.
[51] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming Open Source Project Entry Barriers with a Portal for Newcomers. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 273–284. https://doi.org/10.1145/2884781.2884806
[52] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco A. Gerosa. 2018. Almost There: A Study on Quasi-contributors in Open Source Software Projects. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*. ACM, New York, NY, USA, 256–266. https://doi.org/10.1145/3180155.3180208
[53] Igor Steinmacher, Marco Aurélio Graciotto Silva, Marco Aurélio Gerosa, and David F. Redmiles. 2015. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology* 59 (March 2015), 67–85. https://doi.org/10.1016/j.infsof.2014.11.001

[54] Igor Steinmacher, Christoph Treude, and Marco Gerosa. 2018. Let me in: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects. *IEEE Software* Early Access (2018), 1–1. https://doi.org/10.1109/MS.2018.110162131

[55] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting Developer Productivity One Bot at a Time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. ACM, New York, NY, USA, 928–931. https://doi.org/10.1145/2950290.2983989

[56] Silvia Tamayo-Moreno and Diana Pérez-Marín. 2017. Designing and Evaluating Pedagogic Conversational Agents to Teach Children. *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering* 11, 3 (2017), 491–496.

[57] Stergios Tegos and Stavros Demetriadis. 2017. Conversational agents improve peer learning through building on prior knowledge. *Educational Technology & Society* 20, 1 (2017), 99–111.

[58] N T Thomas. 2016. An e-business chatbot using AIML and LSA. In *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*. IEEE, Piscataway, NJ, 2740–2742.

[59] Christoph Treude and Martin P. Robillard. 2016. Augmenting API Documentation with Insights from Stack Overflow. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 392–403. https://doi.org/10.1145/2884781.2884800

[60] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of Social and Technical Factors for Evaluating Contribution in GitHub. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 356–366. https://doi.org/10.1145/2568225.2568315

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

[62] Alan M Turing. 1950. Computing machinery and intelligence. *Mind* 59, 236 (1950), 433–460.

[63] Simon Urli, Zhongxing Yu, Lionel Seinturier, and Martin Monperrus. 2018. How to Design a Program Repair Bot?: Insights from the Repairnator Project. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '18)*. ACM, New York, NY, USA, 95–104. https://doi.org/10.1145/3183519.3183540

[64] Francisco A. M. Valério, Tatiane G. Guimarães, Raquel O. Prates, and Heloisa Candello. 2017. Here's What I Can Do: Chatbots' Strategies to Convey Their Features to Users. In *Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems (IHC 2017)*. ACM, New York, NY, USA, Article 28, 10 pages. https://doi.org/10.1145/3160504.3160544

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

[66] Jianguo Wang and Anita Sarma. 2011. Which bug should I fix: helping new developers onboard a new project. In *4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '11)*. ACM, New York, NY, USA, 76–79.

[67] Daniel Wilks. 2011. *Statistical Methods in the Atmospheric Sciences*. Academic Press.

[68] Norman Winarsky, Bill Mark, and Henry Kressel. 2012. *The Development of Siri and the SRI Venture Creation Process*. Technical Report. SRI International, Menlo Park, USA.

[69] Vincent Wolff-Marting, Christoph Hannebauer, and Volker Gruhn. 2013. Patterns for tearing down contribution barriers to FLOSS projects. In *12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT '13)*. IEEE, Piscataway, NJ, USA, 9–14. https://doi.org/10.1109/SoMeT.2013.6645669

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

[71] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. 2015. Wait for It: Determinants of Pull Request Evaluation Latency on GitHub. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*. IEEE Press, Piscataway, NJ, USA, 367–371. http://dl.acm.org/citation.cfm?id=2820518.2820564

[72] Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. 2016. Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment? *Information and Software Technology* 74 (2016), 204 – 218. https://doi.org/10.1016/j.infsof.2016.01.004

[73] Alexey Zagalsky, Ohad Barzilay, and Amiram Yehudai. 2012. Example Overflow: Using Social Media for Code Recommendation. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering (RSSE '12)*. IEEE Press, Piscataway, NJ, USA, 38–42. http://dl.acm.org/citation.cfm?id=2666719.2666728

[74] Jie Zou, Ling Xu, Mengning Yang, Meng Yan, Dan Yang, and Xiaohong Zhang. 2016. Duplication Detection for Software Bug Reports based on Topic Model. In *9th International Conference on Service Science (ICSS 2016)*. IEEE Computer Society, Piscataway, NJ, USA, 60–65. https://doi.org/10.1109/ICSS.2016.16

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