Conversational Bot for Newcomers Onboarding to Open Source Projects

James Dominic*, Jada Houser*, Igor Steinmacher**, Charles Ritter*, Paige Rodeghero*
*Clemson University
Clemson, South Carolina
{domini4,jahouse,crritte,prodegh}@clemson.edu
**University of Northern Arizona
Flagstaff, Arizona
igor.Steinmacher@nau.edu

ABSTRACT

This paper targets the problems newcomers face when onboarding to open source projects and the low retention rate of newcomers. Open source software projects are becoming increasingly more popular. Many major companies have started building open source software. Unfortunately, many newcomers only commit once to an open source project before moving on to another project. Even worse, many novices struggle with joining open source communities and end up leaving quickly, sometimes before their first successful contribution. In this paper, we propose a conversational bot that would recommend projects to newcomers and assist in the onboarding to the open source community. The bot would be able to provide helpful resources, such as Stack Overflow related content. It would also be able to recommend human mentors. We believe that this bot would improve newcomers' experience by providing support not only during their first contribution, but by acting as an agent to engage them to the project.

CCS CONCEPTS

• **Software and its engineering → Open source model**.

KEYWORDS

open source software, bot, onboarding, newcomer

ACM Reference Format:
James Dominic*, Jada Houser*, Igor Steinmacher**, Charles Ritter*, Paige Rodeghero*. 2020. Conversational Bot for Newcomers Onboarding to Open Source Projects. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3387940.3391534

1 INTRODUCTION

Onboarding refers to the process of teaching newcomers (new workers) the knowledge and skills needed to succeed in their positions [2]. It serves as a way to integrate new workers into the environment both so they can understand how to succeed in their position but also integrate with their coworkers so they can work effectively as a team. It is during this period that programmers become familiar with a new project, its source code, and its team. While it is considered a necessity, it is also very costly and error prone [6, 22]. In response to the expensive onboarding process, the software engineering research community has responded by experimenting with methodologies, such as pair programming [51], that help overcome the shortcomings of onboarding. Unfortunately, the onboarding techniques described in the literature are largely not applicable to programmers onboarding remotely. OSS projects leverage the coordinated effort from globally distributed stakeholders who build high-quality software [14]. To remain sustainable and to evolve, several projects rely on the onboarding and retention of newcomers. These newcomers serve not only as a workforce to keep the project running but also as a source of innovation [21].

However, attracting newcomers [42] and engaging them [9] are not easy tasks. Previous work shows that the barriers posed during the joining process may lead newcomers to give up on contributing [42]. Joining an OSS project is a complex, multi-stage process and this complexity could push newcomers away from the project [44]. This is because newcomers are expected to learn about the project on their own and can feel lost in the complexity of a project [11]. The barriers related to newcomers' orientation play a key role, especially in keeping the newcomers confident and motivated to continue with the project [43].

There is an increase of bots being developed to support developers in OSS projects [50]. There are many different types of bots that OSS programmers utilize to help them with their work. Content recommendation bots retrieve information from their users and recommend items that may be of interest to those users [34]. They are widely used by services such as search engines, review sites, and online stores in order to encourage users to engage more with the service. Some bots that help programmers by performing repetitive tasks. A typical example of such use of bots in software engineering can be found in software testing [30, 39] and ongoing research in software bug fixing [18, 38]. Finding a suitable project to contribute to can be challenging for newcomers to OSS. NNLRank was a neural network model proposed to recommend projects to newcomers, which they were likely to contribute [26]. Bots have also been developed to recommend experts and reviewers for tasks [8, 35], which may help newcomers to get help if needed.

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3391534

Because of the difficulties that newcomers face when onboarding to new projects, we believe that having a bot that guides the newcomer, offering help when there is no activity, provides resources and recommends experienced developers for assist, the newcomer will feel more confident and will be poised to have a successful contribution. We also believe that, with this bot, newcomers are more likely to stick with a project rather than moving onto the next. Specifically, when the newcomers have the chance to interact with a person from the project, the sense of belonging in a community will increase. We believe this may increase the chances of long-term engagement.

2 BACKGROUND AND RELATED WORK

In this section, we discuss the background and related work of conversational bots, bots in software engineering, and onboarding to open source projects.

2.1 Conversational Bots

The origin of conversational bots (or *chatbots*), dates back to 1950 when Alan Turing proposed that machines could think [47]. Advances in Natural Language Processing (NLP) and Machine Learning (ML) caused (i) a higher usage of bots in several domains; and (ii) partnerships in which computers and humans construct meaning around each other's activities [13]. According to Lebeuf [23], the mainstream adoption of software bots also occurred because of (i) numerous technological breakthroughs; (ii) dominant adoption of both messaging and voice-only platforms; (iii) and the abundance of public APIs and datasets. Bots enhance collaborative work [16] and influence changes in the workplace [24].

Although several popular bots do, in fact, have some language capability, engaging in conversations is not required for software bots [23, 31]. More specifically, *chatbots* stand out from software bots because of their ability to communicate with users through human language. Over the last few years, technological enterprises have developed bots as intelligent personal assistants, such as Apple's Siri [52] and Google Assistant [41], using conversational interfaces to automate personal tasks for users.

Thousands of bots perform specific tasks in a narrow domain of expertise [10]. For example, chatbots have been used for education [19], focusing on students' engagement [4, 5, 15], self-guided learning [36], course advising [20], and tutoring [46] .

2.2 Bots in Software Engineering

Recently, bots are also becoming commonplace in software engineering, supporting social and technical aspects of software development [25, 45]. For example, Matthies et al. [28] proposed a Slack bot to support agile retrospectives aiming to enhance the development process. From a technical perspective, Urli et al. [48] introduced *Repairnator*, a program repair bot that continuously monitors bugs during Continuous Integration (CI), and then tries to fix them by submitting a pull request. In addition, Ren et al. [37] designed an approach to identify duplicated code changes in forks early. Still, Wyrich and Bogner [53] proposed the *Refactoring-Bot*, a bot that automatically refactors the code to remove code smells.

More related to our vision, Cerezo et al. [8] developed a bot aiming to recommend software artifact experts based on the source code repository history. Developers can ask who is an expert in a given file, method, or package, and the bot answer with a list of people created by a recommending system. Peng and Ma [35] analyzed how the developers perceive *mention bot*, a bot that recommends pull request reviewers based on previous interactions with GitHub. Xu et al. [54] also proposed a recommendation bot, *AnswerBot*, which automatically generates an answer to a technical problem mining answers from Stack Overflow. Abdellatif and Shihab [1] designed *MSRBot*, also created a bot that answers developers' questions based on data from software repositories, including questions about developers responsible for commits, number of bugs fixed by a developer, and the date of broken commits.

Although it is possible to notice a growing body of knowledge in the topic, the bots that are available to support software development tasks are usually designed to serve a specific goal. Still, many of them miss the conversational aspect, acting proactively given an external event. More importantly, even with some bots aiming to recommend experts, answers, and reviewers, the newcomer support is still a missing piece in this territory. We advocate that a conversational bot (chatbot) that offers support to newcomers by helping them throughout their onboarding process would play the role of a mentor, recommending projects, artifacts, experts, and to keep track of their evolution.

2.3 Onboarding to Open Source Projects

The onboarding of newcomers in OSS projects has been widely studied [29, 49]. Among the studies that focus on newcomers to OSS projects, some report scripts, paths, and cases of developers successfully joining projects. Other researchers focus on understanding and dealing with the barriers to onboarding newcomers [17, 42, 43]. While joining the project is difficult, retention is also analyzed as a problem in OSS context. Zhou and Mockus [55], for example, found that the *individual's willingness* and the *project's climate* were associated with the odds that an individual would become a long-term contributor. Fang and Neufeld [12] focused on understanding developers' motivation to stay and found that the *initial conditions to participate* did not adequately predict long-term participation, but that *situated learning* and *identity construction* behaviors were positively linked to sustained participation.

Mentorship is usually applied as a way to support the newcomers. However, in OSS, it is not a widely-spread approach to offer formal mentorship programs. Nevertheless, this topic attracted the attention of some researchers. [7, 27, 32] proposed different approaches to identifying and recommending mentors to OSS newcomers, claiming that mentoring would benefit newcomers' onboarding. For example, Canfora et al. [7], proposed a recommending system that mines the project's mailing list and versioning control to find mentors who have already worked on the topic that the newcomer is working on.

Even recent literature has pointed out that the problem of onboarding is still an open challenge. Given the lack to support the onboarding and engagement of newcomers to OSS, and the rise of bots to support software development, our vision adds to this topic by proposing a bot that not only helps to guide newcomers' first steps but also engages the newcomers with community members to promote long-term engagement.

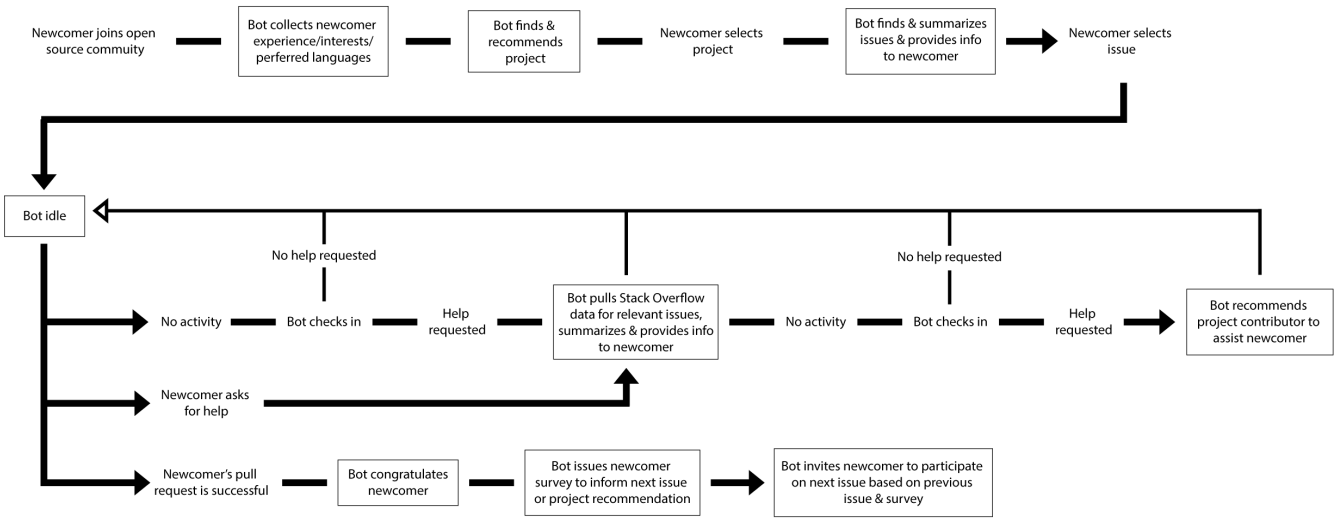

Figure 1: The bot interactions with the OSS newcomer. The various stages and the control flow between them are shown.

3 PROPOSED CONVERSATIONAL BOT

In this section, we discuss the proposed recommendation bot system and the implementation of the system.

3.1 System Overview

We propose to create a conversational bot that could help newcomers with finding open source projects that they can contribute to and guide them throughout the process. For an example of how the bot would interact with the newcomer, see Figure 1. In the next few paragraphs, we will briefly describe the system.

First, when the newcomer initiates a conversation with the bot, the bot would collect information on the newcomer's previous source code contributions, personal interests, and preferred programming languages. The bot would be able to scan for information shared by the newcomers publicly on GitHub (or other OSS platforms). Based upon this information, the bot would suggest OSS projects that the newcomers may find interesting. The bot would provide a brief summary of the present status of the recommended project and the potential tasks that the newcomer may be interested.

Once the newcomer starts working on the project, the bot would go into an idle mode in which it will not interact unless requested by the newcomer—who may ask questions. The bot may be set to wait for a period of time (potentially a week) before it checks in with the newcomer regarding their progress, and offer support. However, the newcomer would be able to interact with the bot any time. The bot would help newcomers with specific programming challenges they face while working towards a successful contribution. The newcomer would be able to ask the bot for help or clarifications. The bot will provide answers based on content mined from Stack Overflow (and potentially other sources), summarize previous related solutions and send relevant links and documents to the newcomer. The bot would go back to an idle state once it has helped the newcomer. It could also check in for a second time if no progress is made in a specific time duration (again, this could be a week's worth of time).

If needed, the bot would also be able to help connect the newcomer with an experienced programmer (someone who has already worked on the current project and has multiple successful contributions). If a connection is made between and experience programmer and newcomer, the bot would send the experienced programmer a summary of the specific issues the newcomer was facing.

Once the newcomer has a successful contribution, the bot would provide the newcomer with a survey so that it could recommend new issues based on the newcomer's experience. However, if desired, the bot could also recommend a different open source project to work on. Contributor disengagement is a costly and critical issue as it can affect the sustainability of a project [40]. In the future, once we are able to build such a bot, we will study the effects of the bot on newcomer engagement.

3.2 Technical Design

The bot's core features would employ methods in Natural Language Processing, specifically in Natural Language Understanding (NLU) and text summarization. In order to implement these features, we plan to incorporate a number of tools. These tools include Rasa Open Source and the GitHub and Stack Exchange APIs.

Rasa Open Source is an open source framework for building conversational bots. It is capable of facilitating the Natural Language Understanding (NLU) process needed for the bot to engage in conversation with the newcomer. GitHub's REST API would allow access to the newcomer's public GitHub information and provide

the means to search site-wide for projects and issues relevant to the newcomer's preferences. The Stack Exchange API provides access to content such as questions, answers, and comments from Stack Overflow as well as other Stack Exchange community QA websites. Thus enabling the bot to collect content relevant to the newcomer's issue for text summarization.

3.3 Conversational Bot and Newcomer Dialog System

The bot could be integrated with collaboration software such as Slack, thereby eliminating the need to develop an interface from scratch. By sending messages to the bot, the newcomer would be able to express and update project interests, receive recommendations and summaries of projects and issues, request assistance, and complete check-in and post-contribution surveys.

4 CURRENT STATE OF DEVELOPMENT

We have started to build a dialogue corpus. The data set currently contains 20 individual conversations between two professional programmers of varying experience levels in a pair programming setting. The programmers were introduced to a new software project and work together to fix source code bugs. Analyzing this data will help us better understand the type of questions which arise from programmers onboarding onto a new project. This dialogue corpus, together with other data will be used for training the bot.

5 CURRENT RESEARCH LIMITATIONS

One research limitation is the size of the data that is needed for training the conversational bot. We have a relatively small dialogue corpus created with conversations between newcomers in a pair programming setting [3][33]. This might impact the accuracy we can achieve while training the conversational bot to work in the specific context. Collecting more data is time consuming and expensive. Unfortunately, we believe that the data collection process must continue until the data set increases with significantly more dialogue or we must rely on an artificial dialogue corpus.

6 CONCLUSION

Our paper proposes a new conversational bot for OSS project newcomers. The vision is that the bot would be able to recommend open source projects, issues from proposed projects that could be of interest, provide helpful resources, and connect the newcomer to human assistance (expert programmers that have had previous successful contributions on the same project). In this paper, we first cover the related work. We then introduce a system prototype for the development of the conversational bot. We provide the steps we believe are necessary for the development of the bot. Finally, we outline some of the research barriers that must be over come to be able to provide a bot that is able to interact with newcomers seamlessly. We believe that this type of conversational bot would increase the retention rate of open source contributors, both in the open source community and to specific open source projects.

REFERENCES

[1] Ahmad Abdellatif and Emad Shihab. 2019. MSRBot: Using Bots to Answer Questions from Software Repositories. *arXiv preprint arXiv:1905.06991* (2019).

[2] Talya N Bauer and Berrin Erdogan. 2011. Organizational socialization: The effective onboarding of new employees. (2011).

[3] Yoshua Bengio, Yann LeCun, et al. 2007. Scaling learning algorithms towards AI. *Large-scale kernel machines* 34, 5 (2007), 1–41.

[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*. ACM, 63–68.

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

[6] Jeffrey Bonar and Elliot Soloway. 1983. Uncovering principles of novice programming. In *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM, 10–13.

[7] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 2012. Who is Going to Mentor Newcomers in Open Source Projects? *(FSE '12)*. ACM, New York, NY, USA, Article 44, 11 pages. https://doi.org/10.1145/2393596.2393647

[8] Jhonny Cerezo, Juraj Kubelka, Romain Robbes, and Alexandre Bergel. 2019. Building an Expert Recommender Chatbot *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 59–63. https://doi.org/10.1109/BotSE.2019.00022

[9] Jailton Coelho and Marco Tulio Valente. 2017. Why modern open source projects fail. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 186–196.

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

[11] Nicolas Ducheneaut. 2005. Socialization in an Open Source Software Community: A Socio-Technical Analysis. *Computer Supported Cooperative Work* 14, 4 (Aug. 2005), 323–368. https://doi.org/10.1007/s10606-005-9000-1

[12] Yulin Fang and Derrick Neufeld. 2009. Understanding Sustained Participation in Open Source Software Projects. *Journal of Management Information Systems* 25, 4 (April 2009), 9–50. https://doi.org/10.2753/MIS0742-1222250401

[13] Umer Farooq and Jonathan Grudin. 2016. Human-computer integration. *interactions* 23, 6 (2016), 27–32.

[14] Brian Fitzgerald. 2006. The transformation of open source software. *MIS quarterly* (2006), 587–598.

[15] 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.

[16] 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*. ACM, 51–56.

[17] Carlos Jensen, Scott King, and Victor Kuechler. 2011. Joining Free/Open Source Software Communities: An Analysis of Newbies' First Interactions on Project Mailing Lists *(HICSS '11)*. IEEE Computer Society, Washington, DC, USA, 1–10. https://doi.org/10.1109/HICSS.2011.264

[18] Kishore Karnane and Corey Goss. 2015. Automating root-cause analysis to reduce time to find bugs by up to 50%. *Cadence Design Systems, Tech. Rep* (2015).

[19] Alice Kerry, Richard Ellis, and Susan Bull. 2008. Conversational agents in E-Learning. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 169–182.

[20] 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.

[21] Robert E. Kraut and Paul Resnick. 2012. *Building Successful Online Communities: Evidence-Based Social Design*. The MIT Press. http://www.worldcat.org/isbn/0262016575

[22] Adriaan Labuschagne and Reid Holmes. 2015. Do onboarding programs work?. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*. IEEE, 381–385.

[23] Carlene R Lebeuf. 2018. *A taxonomy of software bots: towards a deeper understanding of software bot characteristics*. Ph.D. Dissertation.

[24] 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*. European Society for Socially Embedded Technologies (EUSSET).

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

[26] Chao Liu, Dan Yang, Xiaohong Zhang, Baishakhi Ray, and Md Masudur Rahman. 2018. Recommending GitHub Projects for Developer Onboarding. *IEEE Access* 6 (2018), 52082–52094.

[27] Yuri Malheiros, Alan Moraes, Cleyton Trindade, and Silvio Meira. 2012. A Source Code Recommender System to Support Newcomers *(COMPSAC '12)*. IEEE, Los Alamitos, California, USA, 19–24. https://doi.org/10.1109/COMPSAC.2012.11

[28] Christoph Matthies, Franziska Dobrigkeit, and Guenter Hesse. 2019. An Additional Set of (Automated) Eyes: Chatbots for Agile Retrospectives *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 34–37. https://doi.org/10.1109/BotSE.2019.00017

[29] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. 2002. Evolution Patterns of Open-source Software Systems and Communities *(IWPSE '02)*. ACM, New York, NY, USA, 76–85. https://doi.org/10.1145/512035.512055

[30] Bao N Nguyen, Bryan Robbins, Ishan Banerjee, and Atif Memon. 2014. GUITAR: an innovative tool for automated testing of GUI-driven software. *Automated software engineering* 21, 1 (2014), 65–105.

[31] Elahe Paikari, JaeEun Choi, SeonKyu Kim, Sooyoung Baek, MyeongSoo Kim, SeungEon Lee, ChaeYeon Han, YoungJae Kim, KaHye Ahn, Chan Cheong, and Andre van der Hoek. 2019. A Chatbot for Conflict Detection and Resolution *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 29–33. https://doi.org/10.1109/BotSE.2019.00016

[32] Sebastiano Panichella. 2015. Supporting newcomers in software development projects *(ICSME 2015)*. IEEE, 586–589. https://doi.org/10.1109/ICSM.2015.7332519

[33] Meenal J Patel, Alexander Khalaf, and Howard J Aizenstein. 2016. Studying depression using imaging and machine learning methods. *NeuroImage: Clinical* 10 (2016), 115–123.

[34] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[35] Zhenhui Peng and Xiaojuan Ma. 2019. Exploring how software developers work with mention bot in GitHub. *CCF Transactions on Pervasive Computing and Interaction* 1, 3 (01 Nov 2019), 190–203. https://doi.org/10.1007/s42486-019-00013-2

[36] 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*. ACM, 911–918.

[37] Luyao Ren, Shurui Zhou, Christian Kästner, and Andrzej Wąsowski. 2019. Identifying Redundancies in Fork-based Development. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 230–241.

[38] Bilyaminu Auwal Romo and Andrea Capiluppi. 2015. Towards an automation of the traceability of bugs from development logs: a study based on open source software. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 33.

[39] Nurul Husna Saad and Normi Sham Awang Abu Bakar. 2014. Automated testing tools for mobile applications. In *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*. IEEE, 1–5.

[40] Pratyush N Sharma, John Hulland, and Sherae Daniel. 2012. Examining turnover in open source software projects using logistic hierarchical linear modeling approach. In *IFIP International Conference on Open Source Systems*. Springer, 331–337.

[41] Nick Statt. 2016. Why Google's fancy new AI assistant is just called'Google'. *Retrieved March* 21 (2016), 2017.

[42] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*. ACM, 1379–1392.

[43] 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*. ACM, 273–284.

[44] Igor Steinmacher, Marco Aurélio Gerosa, and David Redmiles. 2014. Attracting, onboarding, and retaining newcomer developers in open source software projects. In *Workshop on Global Software Development in a CSCW Perspective*.

[45] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting Developer Productivity One Bot at a Time *(FSE 2016)*. ACM, New York, NY, USA, 928–931. https://doi.org/10.1145/2950290.2983989

[46] Silvia Tamayo-Moreno and Diana Pérez-Marín. 2017. Designing and Evaluating Pedagogic Conversational Agents to Teach Children. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering* 11, 3 (2017), 488–493.

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

[48] 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*. ACM, 95–104.

[49] Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. 2003. Community, joining, and specialization in open source software innovation: A case study. *Research Policy* 32, 7 (2003), 1217–1241.

[50] Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. 2018. The Power of Bots: Characterizing and Understanding Bots in OSS Projects. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 182 (Nov. 2018), 19 pages. https://doi.org/10.1145/3274451

[51] Laurie Williams, Anuja Shukla, and Annie I Anton. 2004. An initial exploration of the relationship between pair programming and Brooks' law. In *Agile Development Conference, 2004*. IEEE, 11–20.

[52] Norman Winarsky, Bill Mark, and Henry Kressel. 2012. The Development of Siri and the SRI Venture Creation Process. *SRI International, Menlo Park, USA, Tech. Rep* (2012).

[53] Marvin Wyrich and Justus Bogner. 2019. Towards an Autonomous Bot for Automatic Source Code Refactoring *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 24–28. https://doi.org/10.1109/BotSE.2019.00015

[54] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: automated generation of answer summary to developersź technical questions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 706–716.

[55] Minghui Zhou and Audris Mockus. 2015. Who Will Stay in the FLOSS Community? Modelling Participant's Initial Behaviour. *IEEE Transactions on Software Engineering* 41, 1 (2015), 82–99. https://doi.org/10.1109/TSE.2014.2349496