RESEARCH

Being a Mentor in Open Source Projects

Igor Steinmacher[1,3*], Sogol Balali[2], Bianca Trinkenreich[3], Mariam Guizani[2], Daniel Izquierdo-Cortazar[4], Grizelda G. Cuevas Zambrano[5], Marco Aurelio Gerosa[3] and Anita Sarma[2]

*Correspondence: igorfs@utfpr.edu.br
[1]Department of Computing, Universidade Tecnológica Federal do Paraná, Campo Mourao, PR, Brazil
Full list of author information is available at the end of the article

Abstract

A well-known way to help newcomers overcome initial contribution challenges is mentoring. This strategy has proven effective in offline and online communities, and to some extent has been employed in Open Source Software (OSS) projects. Through mentoring, newcomers are trained to acquire the technical, social, and organizational skills they need. Despite the importance of OSS mentors, they are under studied in the literature. Understanding who mentors in OSS projects are, the challenges they face, and the strategies they use can help OSS projects better understand and support mentors' work. In this paper, we investigate the OSS mentors' perspectives by employing a two-stage study. First, we understand the characteristics of the mentors in a large OSS community through a large-scale online survey in the Apache Software Foundation. We found that contributors who are volunteers and less experienced are less likely to take on the role of mentoring. Second, we identify the challenges that mentors face and how they mitigate these challenges through interviews with OSS mentors (n=18). In total, we identified 25 general mentorship challenges and 7 sub-categories of challenges regarding task recommendation. We also identified 13 strategies to overcome these challenges. Our results provide insights for OSS communities, formal mentorship programs like Outreachy, and tool builders who design automated support for task assignment and internship.

Keywords: OSS; mentors; onboarding; challenges; task recommendation; software engineering

1 Introduction

A variety of Open Source Software (OSS) projects strive to become open collaboration communities with low entry barriers by facilitating the onboarding of newcomers [30, 79]. However, previous work shows that newcomers to OSS communities face a large number of barriers [81, 82].

Mentorship is a frequently-adopted strategy to help newcomers overcome the barriers they face in their first steps to contributions [24, 38, 50]. In offline communities, assigning mentors to new members has proven effective at helping them overcome challenges [22]. OSS communities also offer mentoring initiatives for newcomers [24, 73, 81], including well-known and established programs like Google Summer of Code [74]. Through mentoring, newcomers are trained to acquire the technical, social, and organizational skills they need [24, 44, 50, 56]. In this context, a mentor is a peer who was assigned to or who volunteered to support newcomers onboarding a project. Mentors are usually peers who succeeded in overcoming the project challenges themselves and are willing to help others onboard [49].

Past work has focused on developing strategies to automatically recommend mentors [15, 47, 56, 77] and to assess the impact of mentoring [24, 44, 70]. In our previous

work [4], we found that typically OSS mentors are neither formally trained nor paid for such work. Without any formal training, mentors–even if they are technically resourceful–are not aware of the challenges involved in mentoring, especially in getting a newcomer started on their tasks. Identifying a starter tasks that is not too complex, yet challenging to motivate the newcomer and that matches their experience is nontrivial [77].

In this paper, we extend our previous works [4, 5] by investigating the characteristics of mentors in a large OSS organization and by providing a comprehensive view of the challenges they face and the strategies they employ. Scientifically identifying challenges and strategies that mentors adopt to support newcomers can help OSS communities, mentors, and researchers create better tools and processes to support mentors. Therefore, we aim to answer the following research questions.

RQ1. What are the characteristics of OSS mentors?
RQ2. What are the mentorship challenges that mentors face?
RQ3. What are the specific challenges and strategies for recommending tasks to newcomers?

To achieve our goal, we employed a survey (n=624) with contributors to the Apache Software Foundation (ASF), one of the largest open source communities. Second, we interviewed 18 mentors from a diverse set of projects to uncover challenges that mentors face and the strategies they employ to mitigate those challenges.

In summary, this paper contributes to the literature by: (a) characterizing the contributors who play a mentorship role, (b) describing 25 challenges mentors face, (c) investigating further the challenges they face when recommending tasks to newcomers, and (d) identifying strategies that mentors employ. Our results inform the broader OSS community on how to better support mentors and tool builders through specific strategies that can used to support task recommendation.

2 Method

To answer our research questions, we designed a study that comprised two stages. Stage 1 provides an understanding of the characteristics of mentors in a large organization (RQ1), while Stage 2 investigates mentorship challenges and strategies from the point of view of the mentors (RQ2 & RQ3). Figure 1 depicts the overview of the two stages in our research method, which we discuss next.

For Stage 1, we employed a survey in collaboration with the Apache Software Foundation (ASF). Focusing on a single organization to answer RQ1 helped us in multiple ways. First, it allowed us to better define our population in the context of a single organization and compare and contrast characteristics of mentor with non-mentors. Second, our partnership with the Apache Software Foundation allowed us to reach a large number of respondents (624) across multiple projects. Such a large survey allowed us to obtain data about 175 mentors. Prior to this large-scale survey, we tried to reach a large number of mentors, but it was difficult since their role is often not formally defined. Third, the ASF serves as a relevant case study as it is one of the world's largest OSS foundation with more than 460,000 people involved, with more than 350 projects and initiatives and partners with mentorship programs such as Outreachy and Google Summer of Code. Finally, Apache projects are often

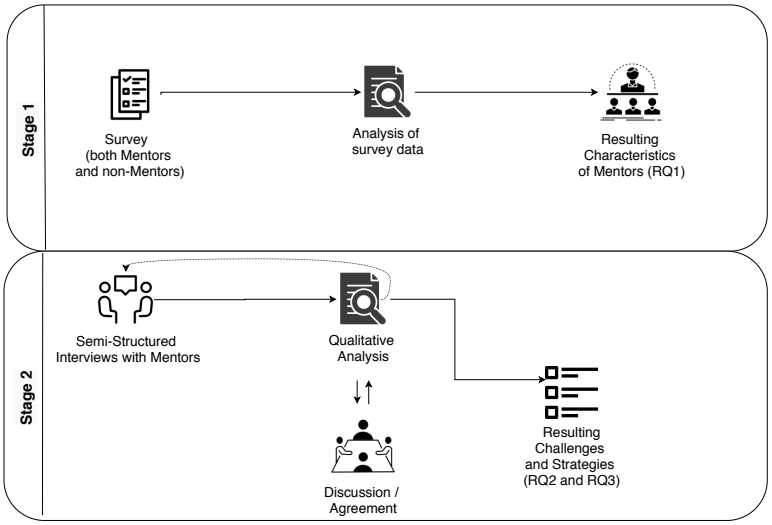

Figure 1 Research Method Overview

studied in scientific research, which allows our work to be placed in the context of existing research [16, 17, 35, 41, 54].

For Stage 2, we conducted in-depth interviews with mentors. For this stage, we interviewed mentors from the ASF and also included mentors from several other OSS communities to provide a broad view of the mentors' challenges and strategies.

2.1 Stage 1 - Study Planning

We defined the survey's target population as any contributor in the community to be able to compare mentors with non-mentors. The survey comprised 25 questions, including demographic questions; their initial and current roles; how often the respondents contribute to the projects; Compensation (paid/unpaid); English proficiency; and if the respondents had a mentor when they started. All the questions were optional to increase the response rate [61]. The survey design was in collaboration with the ASF and underwent several iterations with feedback from the D&I committee of the ASF. The survey was sandboxed with the research team and piloted with the D&I committee members.

The recruitment was based on sending 7010 direct e-mails to the community's committers, D&I lists, and posting in social networks. The survey was available between January 1 and February 24, 2020. We received 624 survey responses resulting in a response rate of 8.5%, considering the total community size of 7,500 committers in the ASF.

2.2 Stage 1 - Data Analysis

To characterize mentors, we used the survey question about how often the contributor performed mentorship activities. This question had four possible answers: Never, Rarely (less than once a month), Sometimes (more than once a month), and Often (once a week or more). From the 624 respondents, 65 did not answer this questions. We considered a respondent as a mentor if the options "Sometimes" or

"Often" were marked, and not a mentor if the options "Never" or "Rarely" were marked. Using this filter, we found 175 mentors and 384 non-mentors.

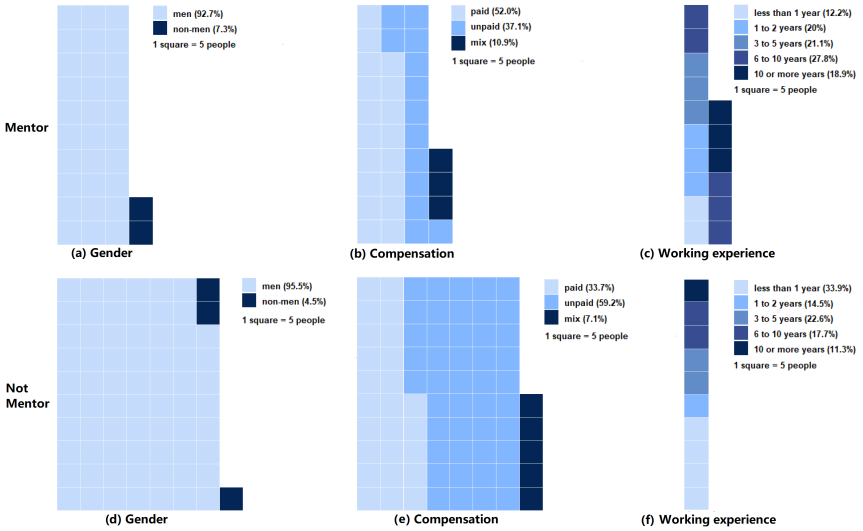

Figure 2 Demographics of Survey Participants who were mentors (top row) and those who are not mentors (bottom row).

Figure 2 presents the demographics of the mentors and the non-mentors. When considering mentors, a large majority were men (92.7%), experienced in the ASF (only 12.2% have less than 1 year experience), and had some component of their work paid (62.9% when combining those fully and partially paid). There are only slight differences in the demographics of the non mentors. Non mentors included 95.5% of contributors who were men and had similar distribution of work experience and compensation structure.

We used regression modeling techniques to identify demographics that positively or negatively influence the likelihood of a contributor being a mentor. Table 1 presents the different values of the dependent variables we use.

Table 1 The categorical dependent variables, respective levels and possible values they could assume

Dependent Variables	Compensation	Experience	Had Mentor	Age	Gender	Education
Possible Values	Paid	<1 year	Yes	<25 years	woman	No formal education
	Unpaid	1-2 years	No	25-34	man	High school
	Mix of Paid and Unpaid	3-5 years		35-44	non binary	Technical training
		6-10 years		45-54	prefer to self describe	Under graduate
		>10 years		55-64		Master's degree
				>65 years		Ph.D.

To check for collinearity among the explanatory variables in the regression models, we first checked for Variable Inflation Factors (VIF). A recommended practice is to remove any variables in the final model that have VIF scores > 5 [31]. Based on the VIF scores, we selected the following variables to be part of the model: com-

pensation, had-mentor, experience, and gender. The VIF scores of these variables were < 2, signifying low collinearity among the variables.

In this paper, we report all analysis results at $\alpha < 0.05$. We used the R statistical software to perform our analysis.

2.3 Stage 2 - Study Planning

Our goal in Stage 2 was to find challenges that OSS mentors face and the strategies they employ to mitigate them. We recruited mentors from the Stage 1 survey who said they would be available for a follow-up interview. Additionally, in order to get a broader understanding of mentors' challenges, we also recruited mentors from several other OSS communities using convenience sampling and snowballing techniques. Specifically, we sent out recruitment emails to two OSS contributors who we knew to be mentors. Additionally, at the end of each interview, we asked participants to recommend other mentors we could contact.

In total, we interviewed 18 experienced OSS mentors, 10 from several OSS projects (P1-P10) and eight from the ASF community (P11-P18). Out of these eighteen participants, 11 self-identified as men, six as women, and one preferred not to disclose. All of our interviewees had a minimum of 1-2 years of experience in the OSS project they are currently in. We kept interviewing until we came to an agreement that no new challenges were identified for at least 2 interviews. According to Strauss and Corbin [87], sampling can be discontinued once the collected data is considered sufficiently dense and data collection no longer generates new information. Table 2 present the demographic information of the interviewees.

Table 2 Stages 2 and 3 - Demographics for the Interviewed Mentors

Participant ID	Gender	Years in OSS	OSS Project
P1	Man	3-5	Apache Lucene, Solar
P2	Man	3-5	Gnome
P3	Man	3-5	RedHat
P4	Man	1-2	RedHat
P5	Man	6-10	Gnome
P6	Woman	Over 10	Linux Kernel, Apache Spark
P7	Man	3-5	Not mentioned
P8	Man	Over 10	Linux Kernel
P9	Man	6-10	KDE
P10	Woman	3-5	Open Hatch
P11	Man	Over 10	ASF
P12	Woman	3-5	ASF
P13	Woman	Over 10	ASF
P14	Woman	Over 10	ASF
P15	Prefer not to say	Over 10	ASF
P16	Man	3	ASF
P17	Man	3-5	ASF
P18	Woman	3-5	ASF

2.4 Stage 2 - Data Collection

We used semi-structured interviews, which consisted of a mixture of open-ended and specific questions that were designed to elicit foreseen and unexpected information types [71]. In this kind of interviews, the questions are planned, and we seek to answer them, but they are not necessarily asked in the same way or order as they are listed [66]. We designed our interview script according to the literature recommendations [66, 71]. Before interviewing the participants, we conducted four

pilot interviews with Ph.D. students who had experience working in industry or OSS environments to validate the script and confirm whether the interview would fit in a 1-hour time slot. The pilot participants answered all the interview questions and provided us feedback about the flow of the script. The results of these pilot interviews were discarded. We also analyzed the questions and answers to ensure that they provided data that would answer our research questions.

Two researchers ran the semi-structured interviews, which lasted around 40-minutes: one researcher led the interview while the other observed and took notes. All interviews were remotely undertaken using Skype, Google Hangouts, or phone. With participant consent, we audio-recorded all interviews, which were transcribed for the analysis. Before each interview, we shared a consent form with the participants asking for their agreement. After the interview, we thanked our participants and compensated them with a gift card or a equivalent value in donation to the OSS project/organization of their choice.

2.5 Stage 2 - Data Analysis

Each interview was manually transcribed, and then three authors employed a card sorting approach [76] to analyze the data. They started by unitizing each interview into individual cards and applied open coding to classify strategies and challenges. During five weeks, the partial results were discussed and validated on a weekly basis with three more experienced authors. The whole process was conducted using continuous comparison [87] during the coding sessions and negotiated agreement [32] (as a group). In the negotiated agreement process, the researchers discussed the rationale they used to apply particular codes and reach consensus on which code should be applied for a given excerpt [29, 32].

We found a total of 25 challenges as presented in Table 4. Mentors from the diverse set of OSS communities (P1-P10) reported 19 of these challenges. Mentors from ASF (P11-P18) mentioned 7 challenges that were common to these challenges and 6 new types of challenges.

3 Results

Our study reveals characteristics of mentors (**RQ1**), challenges they face in mentorship activities (**RQ2**), and specific challenges and strategies for recommending tasks for newcomers (**RQ3**). In this section, we report our findings per research question.

3.1 Characteristics of OSS mentors (RQ1)

In our survey, 559 individuals answered the mentorship question, out of which 175 respondents indicated they play a mentorship role at least once a month. We used a linear regression model to answer RQ1 as discussed in Section 2.2.

Table 3 presents the result of the regression model. The dependent variables "Compensation" where its value is *unpaid* and "Experience" *level* < 1 *year* had negative estimates and were significant ($p < .05$). The negative coefficients indicates that an increase in these factors is linked with a decrease in the likelihood of the contributor with those characteristics to be a mentor. Based on our data, we conclude that **contributors are less likely to be a mentor when they volunteer (not compensated) and have less than 1 year of experience**.

In our dataset, proportionally both women and men had similar distribution of being mentors—31% (153 out of 495 men are mentors) as compared to 32% of all women who serve as mentors (12/28). Indeed, we did not find statistical significance for the impact of gender on being a mentor. However, we highlight that we only have a few women in our sample—5% (28 out of 559), which might impact our analysis. (We do not include those individuals who preferred to not state their gender in this analysis). This proportion of women in our survey respondents concurs with the overall proportion of women (5.2%) in ASF found earlier in [72].

Table 3 The linear regression model with standardized coefficients showing how different factors influence an OSS contributor to be a mentor. The level of statistical significance is indicated with asterisks: (*) for p ¡ .05, (**) for p ¡ .01, and (***) for p ¡ .001

Coefficients	**Estimate**	**Pr(>—z—)**
(Intercept)	1.2743	0.0341 (*)
Partially Paid Contributor (provides paid and unpaid contributions)	1.5077	0.1817
Volunteer Contributor (all contributions are Unpaid)	-0.8696	0.0299 (*)
Contributor who had a mentor	0.1299	0.7418
Experience as Contributor between 1 to 2 years	-0.5238	0.4328
Experience as Contributor between 3 to 5 years	-0.7232	0.2578
Experience as Contributor between 6 to 10 years	0.1252	0.8488
Experience as Contributor less than 1 year	-1.5859	0.0153 (*)
Identified the gender as woman	1.0728	0.204

3.2 Mentorship challenges (RQ2)

In this paper, we extend our previous work [4], in which we qualitatively analyzed data collected from interviews with mentors. We found 19 categories of challenges that affect mentors classified as:

- *Personal challenges* – related to personal characteristics of mentors;
- *Interpersonal challenges* – related to the relationship between community and mentees;
- *Process challenges* – imposed by the organization or by internal procedures or practices;
- *Technical challenges* – directly related to or caused by technology, including frameworks, programming languages, and/or tooling used in the project.

In the present study, we extended Balali et al. [4] by interviewing mentors from the ASF. We found 13 categories of challenges in this new dataset, 7 that were already previously identified and 6 new categories. We classified the new challenges using the above schema. Table 4 presents the aggregated results. P1-P10 are participants from Balali et al. [4] and P11-P18 are the ASF participants.

3.2.1 Process challenges

Table 4 presents the five process related challenges that mentors face; with three that come solely from the ASF interviewees and identified with a $*$ in the table.

The challenge DIFFICULTY IN IDENTIFYING APPROPRIATE TASKS FOR NEWCOMERS was highly pointed as a challenge for mentors. According to P3, *"to keep them [the newcomers] engaged you need [. . .] to pick a task that is appropriate for them. . . , which can be a challenge for mentors."* When a newcomer's background and goals are unclear, it can be difficult for the mentor to point them to a specific task. Choosing a task was also previously identified as challenging for newcomers

Table 4 General Challenges Faced by Mentors. Each participant could mention multiple challenges. Cells marked with (*) represent the new challenges we found in the present study and the participants who mentioned them.

Category	Challenge Name	Challenge ID
Process	Difficulty in identifying appropriate tasks for newcomers	Pro1
	Not having a formal procedure for introducing the community (P11, P17)	Pro2
	(*) Lack of established governance with processes and rules (P11, P12, P14, P15)	Pro3
	(*) Lack of process to shift mentors (P13)	Pro4
	(*) Mentors do not participate in decisions to promote people (P14)	Pro5
Technical	Difference in the devices that mentors and mentees use	T1
Personal	Handling a large number of mentees	Per1
	Difficulty in switching context	Per2
	Difficulty in time-management (P12, P17)	Per3
	Difficulty in managing different accounts	Per4
	(*) Not getting paid to be a mentor (P18)	Per5
Interpersonal	Adjusting interaction style to different mentee personalities	I1
	Difficulty guiding mentees who are resistant to coaching	I2
	Providing constructive feedback based on the mentee's background	I3
	Convincing people to start small rather than big	I4
	Ensuring that the mentees finish their work	I5
	Difficulty in creating an inclusive community (P17)	I6
	Difficulty to keep the mentees engaged (P16)	I7
	Cultural differences	I8
	Communication issues related to time zone and place (P14, P16, P18)	I9
	Lack of English language skills (P13)	I10
	Lack of mentor's interpersonal skills (P15, P17)	I11
	Harsh project atmosphere	I12
	(*) Difficulty to manage a financially mixed team (paid and volunteers) (P11)	I13
	(*) Difficulty to track the mentees' progress (P15, P18)	I14

as well [80]. Due to the identified relevance of this challenge, we went deeper on it in RQ3.

In addition, if processes are unclear, the mentor must figure out how to get their mentees the information they need. NOT HAVING A FORMAL PROCEDURE FOR INTRODUCING THE COMMUNITY was reported as a challenge by P9, who stated that

> "[. . .] the challenges I have faced are related to how to decide which part of the community to introduce first to the students. It is not totally clear in ¡project-name¿ since we have many processes and don't have a formal procedure for the introduction."

Mentors also reported they miss a *"definitive guide to participate"* (P11) or a *"contribution path that helps people progress through the project"* (P17).

ASF participants also mentioned a LACK OF ESTABLISHED GOVERNANCE WITH PROCESSES AND RULES leads to misinformation and behavior issues. When having *"unstated rules"* (P14) or when *"things are not documented, there are no rules"* (P11). Associated to the the rules, *"there need to be consequences for action when people derail in the mailing lists"* (P12). Besides rules and consequences, mentors complain about the lack of a standard and documented guideline with the main regular processes that contributors need to follow. When not having a documented process to follow, mentors need to remember *"how we did last time"* (P15) or each one can decide *"what is the best way to do it"* (P15). Moreover, there is a LACK OF PROCESS TO SHIFT AND SUBSTITUTE MENTORS. Finally, although they *"try*

to influence decisions" (P14), MENTORS DO NOT PARTICIPATE IN DECISIONS TO PROMOTE PEOPLE.

3.2.2 Technical challenges

Only one technical challenge was mentioend by mentors: DIFFERENCES IN THE DEVICES THAT MENTORS AND MENTEES USE. When mentors and mentees are not using compatible devices or operating systems, it is hard for a mentor to help resolve a newcomer issue. P2 stated

> "the operating system and distribution my computer is running is very different to what the newcomer is running. If a newcomer has an issue, I try to reproduce it, and I may not have this issue which makes it harder to help."

3.2.3 Personal challenges

We identified five personal challenges that impact mentors, with one specifically about the ASF (marked by $*$ in Table 4). The challenges relate to their ability or lack of ability to manage the responsibilities that come along with mentorship. *Handling a large number of mentees* can be overwhelming, as stated by P6: *"I really wish it was easier to deal with a lot of people."* This challenge is related to scheduling, which also creates DIFFICULTY IN SWITCHING CONTEXT between helping mentees and doing their own work. P10 explained that *"if you are not actively focusing your attention on [your mentee] continuously, context switching can be difficult between doing my work and helping them with theirs."* As a part of mentorship, mentors are expected to complete their own work and be available to help their mentees. Three participants from previous work and two from ASF mentioned that DIFFICULTY IN TIME-MANAGEMENT can be challenging, since mentors must choose how to allocate their time to the project, sometimes weighting different activities, such as working on code, mentoring, and reviewing. Mentors feel *"overcommitted"* (P12), as mentorship is *"time-consuming"* (P17) and finding *"time is the hardest thing"* (P12).

Mentors can also encounter problems in aligning their schedule with newcomers, as mentioned by P4: *"being able to contact them [the newcomers] and give feedback was sometimes difficult."* Finally, DIFFICULTY IN MANAGING DIFFERENT ACCOUNTS was mentioned as a challenge, since *"it's really annoying to have a lot of accounts to keep track of."*

In the present study, we identified one new category of personal challenge of NOT GETTING PAID TO BE MENTOR that can affect the mentor's motivation and availability, as mentioned by one of the interviewed mentor: *"[...] as long as you're not working on the project full time, or you're not getting paid, it would be really difficult to mentor [...] unless we get something in return"* (P18).

3.2.4 Interpersonal challenges

Interpersonal challenges is the category with the highest number of challenges reported by the mentors; twelve already identified in Balali et al. [4] and two new ones from the ASF context. Indeed, social interactions play a key role in mentoring.

First, since people who work in an OSS project come from diverse cultures, CULTURAL DIFFERENCES is challenging for mentors. P8 mentioned *"in some cultures, people get more upset when people criticize their code... which can be tough."*

Moreover, when newcomers and mentors are geographically distant, they do not have the opportunity for face-to-face interaction, which can, for example, inhibit informal communication and reduce trust. Therefore, COMMUNICATION ISSUES RELATED TO TIME ZONE AND PLACE affect the communication process during mentorship, as people usually *"can't talk in person"* (P14) and *"most of the communication happens async on Slack"* (P18). Mentors can feel they *"haven't communicated them [mentees] well enough"* and be disappointed when *"[mentees] write an email and have to wait. Sometimes one day, two days"* (P16).

Also related to communication in global settings, LACK OF ENGLISH LANGUAGE SKILLS was mentioned by P9 as hindering the mentorship process: *"My English is so-so . . . when both parties have difficulties communicating, it is challenging to overcome and we don't have good tools for that."* Having a *"English [that is] not good makes [mentors] afraid to participate in tech discussions"* (P13). The language can also impact reading documentation, as *"if that documentation is not equally legible by everybody who claims to be part of [the community], there is a problem"* (P13).

We also observed that a mentor's inability to interact with newcomers (LACK OF MENTOR'S INTERPERSONAL SKILLS) can greatly impact a newcomer's decision to continue contributing to the project. *"People might not have the skills to be a mentor, or knowing how to teach somebody"* (P17) and *"have patience [with mentees]"* (P15, P17). Mentors frequently highlight the importance of social aspects, as evidenced by P3: *". . . the biggest pitfalls of the mentor are: not being responsive and not engaging in other ways than just coding. These projects are about community effort and more than just the code."*

Mentors also face challenges in adapting to how different types of people learn and take in the information presented to them. Two mentors reported that ADJUSTING INTERACTION STYLE TO DIFFERENT MENTEE PERSONALITIES is a challenge, since mentors are likely to collaborate with diverse people who have unique personalities and working styles, as stated by P9: *"[. . .] you always have to adapt based on each individual newcomer [. . .] one solution doesn't always work for everyone."* Mentors need to understand their mentees and tailor aspects of the coaching to fit them. For a mentor, determining how to be an effective teacher for a mentee can be difficult. Four mentors mentioned DIFFICULTY GUIDING MENTEES WHO ARE RESISTANT TO COACHING. Sometimes mentors are required to face the challenge of teaching newcomers who lack a desire to learn. In this sense, P5 mentioned *"But I still don't know how to help people who don't want to learn.* Also related to coaching, mentors reported that PROVIDING CONSTRUCTIVE FEEDBACK BASED ON THE MENTEE'S BACKGROUND is challenging. Mentors must tailor their comments and criticism to the way a newcomer learns, while taking into account their prior experience and level of self-efficacy. P4 reported that *"being able to understand the student's background and the way they see this stuff and give proper feedback is kinda hard."* Some mentees value feedback, while others may not easily perceive it in a constructive manner.

Moderation is sometimes required for mentors when dealing with newcomers. For example, newcomers who are eager to contribute something relevant to the project tend to start with a task that may be too large or complex for their skills set. CONVINCING PEOPLE TO START SMALL RATHER THAN BIG was reported as a dif-

ficulty, as explained by P6: *"the other challenge is convincing people to start small rather than big because lots of people want to make big changes but I can't help them with those."*. This challenge relates to the process challenge called "difficulty in identifying appropriate tasks for newcomers." Ensuring mentees finish their work was reported as a challenge by P3, who mentioned that *"the biggest challenge is making sure they are working and making sure they will finish the project. Otherwise, it is a fail for the mentor if the mentee doesn't finish."*

Mentors also mentioned a Difficulty to keep the mentees engaged, because *"people come, they come, they fix their bug and they go away."* (P16) Indeed, Pinto et al. [59] showed that a great number of newcomers place a single contribution and do not return to the project. Additionally, Steinmacher et al. [83] showed that many newcomers submit one or more contributions, which are not accepted by the community, and they do not come back.

As the community grows, the diversity of contributors grows in parallel. Inclusion is important for attracting newcomers, as well as retaining them and increasing their productivity [91]. The participants of our study seemed to be aware of this and placed particular emphasis on this challenge. Mentors mentioned the difficulty in creating an inclusive community as a challenge. Mentors try to ensure that newcomers feel comfortable and are not discriminated against. P3 explained, *"It is about the community. There has been a lot of discussion about gender pronouns and this is very important to take into account to make sure the community is inclusive of all, especially for newcomers."* The community can miss diversity also in terms of tenure, when not bringing novice leaders, as stated by P17: *"no one with less than 10 years of experience is well regarded to lead anything at the foundation, which then imposes this chicken egg analogy into diversity and inclusion."*

A frequently mentioned challenge was harsh project atmosphere (mentioned by 8 out of 18 mentors). This challenge affects mentors, since they face difficulty in supporting newcomers who fear disagreements among committers in the community, as stated by P1: *"I may find a patch to be fine and ready to commit but some other committer may look at it and not agree that it is fine."* This is particularly challenging for mentors, since it is largely out of their control.

Difficulty to track the mentees' progress is a new category from the present study. As mentioned by one of the interviewed mentors: *"They [the mentees] do not come out easily. And that is when I had to go in to [help]: Did you do this step? What was the output for this? [...] I'm not so extensive on doing this, but [do] from from time to time. [...]"* (P15). This challenge can be related to a lack of management skills of the mentor on monitoring the progress of remote teams or to lack of attitude of the mentee on reporting pitfalls and asking for help. Mentors and mentees should both have clear responsibilities to know the boundaries of each one's work to mitigate this challenge. Communication hurdles can also be a cause for this challenge, as mentioned *"[Having] the visibility of the work they were doing, because everything was so async"* (P18).

3.3 Challenges in recommending tasks for newcomers (RQ3)

In order to answer this RQ, we go deeper in the challenge Difficulty in identifying appropriate tasks for newcomers (Pro1 from Table 4). The analysis of the interviews resulted in seven sub-categories for this challenge. We also catalogued

thirteen strategies employed to overcome those challenges, which we grouped into five categories. The challenges are shown in Table 5 and discussed as follows.

Table 5 Challenges faced during the process of task recommendation for newcomers in OSS projects

Challenge name	ID	Mentioned by
Challenging tasks can create social fears in the newcomers	Pro1.1	P1, P3, P4, P10
Mentors have to deal with newcomers' lack of holistic understanding about the project and its culture	Pro1.2	P1, P2, P3, P4, P9
Lack of information about how newcomer-friendly a task is	Pro1.3	P1, P10
Difficulty in identifying the complexity of a task	Pro1.4	P7, P8
Difficulty in estimating the amount of time necessary to finish a task	Pro1.5	P1, P8
Lack of friendly tasks available for newcomers	Pro1.6	P2, P4, P5
Lack of available information about newcomer's skills, interest, and expertise	Pro1.7	P2, P4, P7, P9

(Pro1.1) Challenging tasks can create social fears in the newcomers . Due to the socio-technical nature of OSS projects, newcomers may feel fearful of exposing a weakness or failing. P1 noted that *"Usually, because of social fear, they just back off. They think they are not good enough or they don't know enough."* P4 also mentioned that *"the biggest barrier is being afraid of being judged."* P10 similarly stated: *"sometimes newcomers will be shy to ask for help or not actively engaged when not knowing where to start."* Mentors need to deal with these issues when directing newcomers to appropriate tasks.

(Pro1.2) Mentors have to deal with newcomers' lack of holistic understanding about the project and its culture . Projects are a complex socio-technical landscape, and newcomers have contact with isolated parts of the code or tasks. As mentioned by P1, *"when a newcomer comes in, one thing is they don't understand all the moving pieces of the codebase. Even if they get the code, they don't get the impact on the bigger scheme of things."* To make things worse, projects have specific conventions and protocols, and, as mentioned by P2, *"sometimes the people are so used to these conventions, they have a hard time communicating them because they don't realize they exist."*

(Pro1.3) Lack of information about how newcomer-friendly a task is . Often, there is a large number of tasks in the issue tracker—including potentially easy ones—but *"no direct way to spot tasks suitable for newcomers"*. P1 mentioned that *"we don't know if things are suitable for newcomers."* They complain that people who add issues do not add relevant information and tags that would indicate that the issue is simple to fix. Moreover, the tags that indicate easy tasks only appear for tiny and isolated tasks. P10 identified a concern about such tasks for newcomers: *"I dislike having things labeled as 'bite-size' because it may cause someone to skip talking to anyone and just grab something. You miss out on a lot of important social interaction."* According to this mentor, isolation and lack of interaction for newcomers can increase their social fear.

(Pro1.4) Difficulty in identifying the complexity of a task . Some mentors also reported difficulties in estimating a task's complexity. P8 said *"We don't have a good way [to evaluate the difficulty of a task]. (...) There's a lot out there to analyze the complexity of the code, but I've found none of it to be helpful in the ¡project-name¿."* Therefore, developers most often rely on their experience to evaluate a task. However, experts suffer from the "curse of knowledge"—the difficulty seeing

something from an outsider's point of view [67]. This was observed by P7: *"A task may be more involved and technical than we thought initially. Then we may realize a task isn't suitable for someone because it's actually an expert level task."*

(Pro1.5) Difficulty in estimating the time necessary to finish a task. Complementary to the complexity of the tasks, mentors are also concerned about its duration. Usually, mentors do not want to give newcomers long-term assignments and, sometimes, even low complexity tasks can take long. However, determining the time to complete a task is not trivial, even for easy/non-complex tasks. P8 mentioned this difficulty: *"It's hard to estimate the time needed for a task. I'm not good at gauging what is appropriate for a newcomer."* This can be even worse when newcomers' tasks need to be reviewed by other stakeholders for compliance or other reasons, as mentioned by P1:

> "Newcomers have to wait a while sometimes for a review, and it may even take months to get a review because people are busy."

(Pro1.6) Lack of newcomer-friendly tasks available. Sometimes, *there are no easy-enough tasks* available at the moment. Some interviewees reported that:

> "if you're a newcomer and come at a time when there aren't many tasks open for newcomers, ...then the difficulty lies in finding something in a certain application to turn into a newcomer task [P2]."

(Pro1.7) Lack of available information about newcomer's skills, interests, and expertise. Since mentors may lack information to help them assess newcomers' characteristics, they often misjudge newcomers' abilities, as P9 mentioned: *"sometimes you think they can take on more than they, in reality, are capable of."* Mentors often do not have access to a portfolio, or a set of previous work to evaluate the newcomers' abilities, as P4 said *"some [newcomers] are very good but they don't have some work portfolio to show their previous skills."* As P8 explained, "we don't have a good way to match a task with a developer."

3.3.1 Strategies for recommending tasks to newcomers

We identified 13 strategies employed by mentors to alleviate the challenge of recommending tasks for newcomers, as presented in Table 6 and Figure 3. We classified the strategies into five main categories, as described below.

Table 6 Strategies to identify task complexity

Category	Strategy Name	Mentioned by
Identify task complexity	Reproducing the bug	P2
	Comparing the complexity of new bugs with existing bugs in bug tracker	P2
	Tagging the task based on difficulty	P1, P2, P4, P5, P6, P7
	Adding documentation	P2, P4
	Discussing tagging in the conferences	P1

Strategies to identify task complexity Identifying task complexity is challenging even for mentors (Pro1.4). We explicitly asked our participants if they use a tool to help determine a task's difficulty. Eight out of ten participants mentioned that they do not use any tool but reported several strategies as follows.

Reproducing the bug For bug-related tasks, mentors reproduce the error to further understand it. As P2 explained, *"First of all, we reproduce the bug and try*

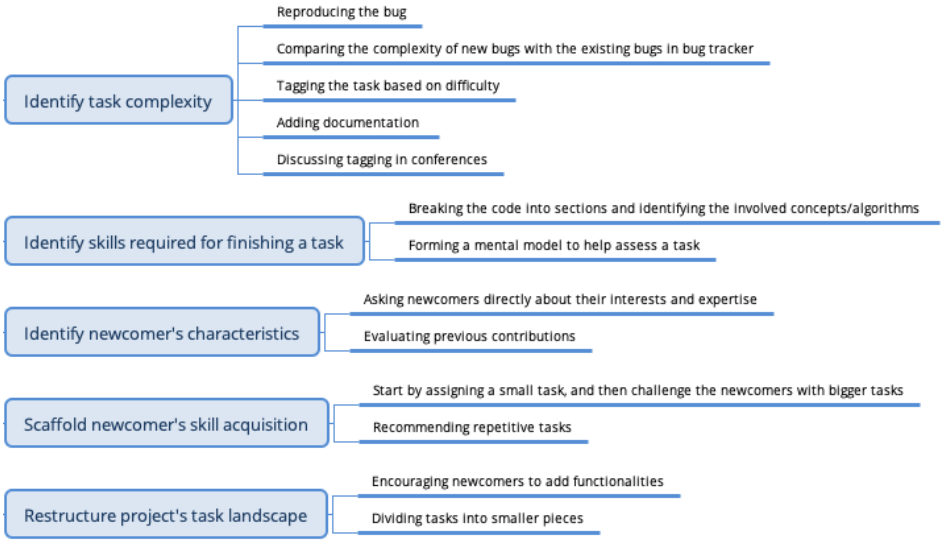

Figure 3 Overall view of the strategies

to see what is causing it and find out what is necessary to fix it. Then we assess whether the skill level is similar to what a newcomer would have to fix it."

Comparing the complexity of new bugs with existing bugs in the bug tracker To further understand task complexity, mentors use previously tagged issues as a scale, comparing the complexity of new tasks with existing ones to find newcomer-friendly tasks, as P2 stated: *"we use, for example, the bug tracker to figure out if a bug is at the same level as other newcomer bugs we have."*

Tagging the task based on difficulty Six mentors reported that, in their projects, tasks are tagged based on difficulty. Tags make newcomer-friendly tasks more visible and identifiable by mentors and newcomers. As P5 also pointed out, *"Mentors are strongly encouraged to tag tasks for newcomers based on complexity (how many concepts does a newcomer need to know, how deep should one's knowledge be)."* In Figure 4, we can see examples of issues labeled as "good first issues." As P4 described, *"we have this tool called Bugzilla, which we are tracking all the bugs. When we file a bug we find to be easy; instead of going directly to fix it, we tag it with a newcomer tag so that we have in our website a list of bugs that are suitable for newcomers."* P6 added that, *"we basically tag things that are good starter issues. If someone asks, we show them these starter tasks list. It's not perfect, but it works."* On the other hand, P7 complemented that newcomers do not want to only work on beginner tasks and that labelling a task "beginner" is not very rewarding to those working on the tasks. This mentor also reported having "medium" and "major" tags, which people can progress through as they go. However, P10 remembered that evaluating task difficulty is complicated: *"You're describing the relationship between the person solving it and the task itself. For one person, it may be easy, and for someone else it may not, it depends on the background. Some tasks are more difficult for more people, so you can say it's difficult overall."*

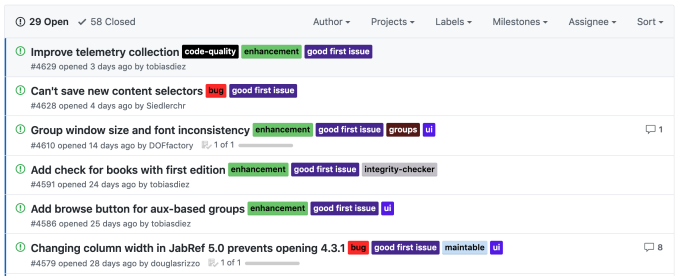

Figure 4 JabRef's issue tracker with labels ("good first issues") to help newcomers identifying appropriate tasks

However, as previously mentioned, the lack of information about how newcomer-friendly a task is (Pro2.3) was frequently cited as a challenge. P10 affirmed that tags might not be enough: *"it can be tough if a project doesn't do a good job grooming tasks. When you label something as a good task for newcomers, it's necessary to check out some things before making that assessment that it's an easy task."*

Adding documentation Sometimes, project members also add information in the issue tracker to help newcomers, as explained by P4: *"we put in the description how we want the contribution to be and which file they should look at. So we are pretty much tagging bugs and creating tutorials on how to fix this specific bug so they can learn the whole process of how to make their contribution."* P4 further explained that the required skills should also be clear: *"You look at the description of the task, you should be able to see if it requires familiarity with C, JavaScript, regular expressions, etc."* P2 further added: *"usually when mentors assign the newcomer tag, we add clarifying info for the newcomer to read... we also provide specific resources for specific projects."*

Discussing tagging in conferences Due to the relevance of tagging tasks in OSS project environments, members of OSS projects hold discussions to improve tagging, as P1 explained: *"Some committers get together at conferences and meetups to discuss ideas to help newcomers when getting involved... We discuss tagging at those conferences."*

Strategies used to identify skills required for finishing a task Besides determining task complexity, mentors and newcomers need to understand the necessary skills to complete a task. We found that mentors usually employ two strategies to identify the skills needed for working on a task, as listed in Table 7.

Table 7 Strategies to identify the skills required to finish a task

Category	Strategy Name	Mentioned by
Identify skills required for finishing a task	Breaking the code into sections and identify the concepts/algorithms involved in that task	P5, P6, P10
	Forming a mental model to help assess a task	P6, P10

Breaking the code into sections and identifying the involved concepts/algorithms Breaking the code into pieces may help to identify required skills, as explained by P5: *"To measure the skills, I go through the code line by line and break it into sections to tell what concepts are needed. I determine what algorithms they would need to know, what hardware they need to know, and what*

coding skills they need to work the bug." However, recognizing the required skill set for finishing a task is not an easy endeavor for mentors, as P10 stated: *"[identifying what skill is required for finishing a task] is a little tough because there may be some unexpected elements."*

Forming a mental model to help assess a task We noticed that our participants reported forming a model of the project structure in their minds, which helped them recommending the appropriate tasks for newcomers. For instance, P6 mentioned *"When I see a task, I can make a mental model of where it may fit."* Mental models are internal representations of the world that help humans understand, describe, and anticipate events and situations [39, 40]. P10 reported that she thinks through a workflow model to consider the required skills and possible locations for a bug.

Strategies to identify newcomer's characteristics Another important aspect of supporting task selection is determining newcomers' characteristics, including but not limited to their interests, expertise, and areas of improvement. Mentors report identifying newcomers' characteristics as challenging (Pro1.7). Eight of our participants mentioned they evaluate newcomers' expertise, and five mentioned they look for what newcomers are interested in before recommending tasks to them. Our interviewees highlighted that they avoid recommending tasks that newcomers do not like, as P3 stated: *"I like to ask them if they would be interested in a task before assigning."* Therefore, it is important that mentors identify newcomers' interests and areas of expertise. From the analysis of the interviews, we found two strategies mentors apply to help identify newcomers' characteristics, as presented in Table 8.

Table 8 Task recommendation strategies to identify the newcomers' characteristics

Category	Strategy Name	Mentioned by
Identify newcomers' characteristics	Asking newcomers directly about their interests and expertise	P1,P3, P4, P6, P7, P9, P10
	Evaluating previous contributions	P4, P9, P10

Asking newcomers directly about their interests and expertise Our participants learn about newcomers' interests and characteristics by directly asking them what they like and what their past experiences involved. P10 mentioned, *"I try to talk to them and help them before sending them off to look at the issue tracker. I gauge their interests and their mindset about the process to find a task for them."* P6 complemented this view: *"I will ask someone what they are excited about and what they think their skills are, and then I'll tell them where to look, and they'll come back with some issues."*

P10 explained that her first question is usually: *"what is your background?"* She also explicitly asks about previous experience with the platform: *"If a person says they haven't worked on GitHub, that lets me know a big part of their first contribution will be getting to know how to use these basic tools. So maybe they will do something like fix a typo, very straightforward."*

Mentors also take into account the initial confidence of newcomers toward a specific task before assigning it to them. When people are initially confident, they associate success with their ability and failure with bad luck [28]. In this sense, P1 stated that, *"If they're confident with their technical skills, we point them towards stuff involving what they know."*

Besides the current skills, mentors also ask newcomers what they want to improve, as P10 explains: *"The two main things are: their [newcomers'] experience and how much they are looking to learn within a specific task. Do you want something easy for you? Or do you want to jump into something above your skill level?"* Mentors report that having had some experience with the task creates a positive impact on their performance and motivation, as P1 stated: *"If someone is good with ZooKeeper interactions, we would point them towards areas that focus on that."*

Evaluating previous contributions Previous contributions provide evidence about newcomers' expertise, as explained by P4: *"If we have some way to see past contributions of the newcomer, you can easily see it is something related to that they have done before."* However, the mentor added that *"Some students are very good, but they don't have some work portfolio."*

Scaffolding newcomer's skill acquisition Mentors frequently mentioned how they recommend a sequence of tasks to onboard newcomers, as presented in Table 9 and described below.

Table 9 Strategies to scaffold newcomers' skills acquisition

Category	Strategy Name	Mentioned by
Scaffold newcomer's skill acquisition	Assigning a small task first and then challenging the newcomers with bigger tasks	P3, P7
	Recommending repetitive tasks	P3
	Letting newcomers choose their tasks	P1, P2, P4, P5, P6, P8, P9

Assigning a small task first and then challenging the newcomers with bigger tasks According to our interviewees, offering newcomers small starter tasks provides mentors the opportunity to evaluate newcomers' skills and interests and support them through the learning curve. Regarding this, P7 stated: *"Sometimes . . . you have to start on the basic tasks and go from there. We see how they are doing and then move forward."* Furthermore, mentors state that assigning small tasks that newcomers can complete keeps them motivated. P3 explained the task *"has to be technically very simple[...] For example, one of the tasks we have for newcomers is modifying strings on the UI, so they get excited about having made that first contribution and see that everyone is using it. Since they face so many other challenges, the first task should not be technical. They have to figure out the tooling and Bugzilla so there is a lot they must overcome."* Mentors adjust the trajectory of tasks based on how they see the newcomers' performance: *"if the task should take about a week and the person finishes in a couple of days, you think 'hmm this may be too easy for them'."*

Recommending Repetitive Tasks Mentors also use a strategy of assigning newcomers repetitive tasks to help them master specific skills and gain confidence before they move to more complex tasks. For example, P3 mentioned that *"I usually choose tasks that are repetitive but not very technical, so maybe modifying many lines of code in the same way. It is not very technical, but then they learn a bit more about the programming language or the API."* P6 complemented this view: *"Usually, its better if they don't have to learn a new skill right away."*

Letting newcomers choose their tasks This was not classified as a strategy, but we want to highlight that some mentors prefer to let newcomers find tasks that suit their expertise and interest, as P5 explained: *"I don't assign tasks. One step of being a contributor is choosing your own task. We use Bugzilla, which lists all*

the bugs and tasks, and newcomers go there and pick something to work on. People come to me when they don't know what to do. Then, I guide them about what is easier and more complex and give them pointers." This view is shared with P1: *"they usually just pick up something they know they would like."* Seven of our interviewees mentioned that they usually identify a subset of tasks and allow newcomers to choose the tasks they are more willing to accomplish.

Restructure project's task landscape There are times during the project life cycle in which no newcomer-friendly task exists [4]; this was mentioned as a challenge by our mentors (Pro1.6). In these cases, mentors apply strategies to restructure the project's task landscape to explore or define newcomer-friendly tasks. We found two strategies that mentors employ to achieve this goal, as can be seen in Table 10.

Table 10 Strategies to restructure the task's landscape

Category	Strategy Name	Mentioned by
Restructure project's task landscape	Dividing tasks into smaller pieces	P2, P3
	Encouraging newcomers to add functionalities	P5

Dividing tasks into smaller pieces Whenever possible, mentors divide tasks into smaller pieces, as P3 explained: *"usually we can divide the tasks based on the project itself and knowledge about specific parts of the project."* This mentor preferred to create tasks related to the user interface: *"There are parts of the projects that are low-level and parts using the UI. The UI parts are less technical and less demanding for a contributor while low-level tasks are more demanding."*

Encouraging newcomers to add functionalities Another strategy reported by our interviewees involves encouraging newcomers to propose and add new functionalities to the project. Regarding this, P5 stated: *"In Gnome To Do, in the beginning, there were 6 tasks which were big and not suited to newcomers. Because I didn't have any easily fixable tasks, I encouraged newcomers to add new functionality as a way to contribute."*

4 Discussion

In the following, we discuss the implications of this study for research and practice in light of our results and related literature.

4.1 Supporting Mentors

Mentors in OSS projects provide valuable guidance and perspective as contributors, which has been shown to be beneficial [4, 25]. However, as our work reveals being a mentor in OSS comes with its own own set of challenges, many of which are outside of the technical realm. Further, as our participants revealed there is no training available for mentors. None of the participants in our study had received any mentoring-related training. Given the challenges that mentors face (Table 4), it is not surprising that contributors were more likely to be mentors when they were being paid to contribute to OSS. We posit that these (compensated) mentors might be taking on this role to help their colleagues from the same company. Another reason for volunteers to be less likely to be mentors can be because mentoring remains "invisible" work [37], not getting the same level of recognition as technical contributions. A case in point are the leadership boards that only account for commits in their calculations. It is likely therefore, that volunteers have to decide where their

contributions can get them the most recognition (recognition is a key driver for participation in OSS [33, 93]) and forgo mentoring activities. Our results highlight the need for large communities to acknowledge and recognize invisible work like mentoring, since it serves as an important bridge for newcomers [22].

When thinking of supporting mentors, we need to recognize that it involves much more than providing technical resources or help. In fact, the majority of the challenges reported by participants were non-technical in nature. Thus, it is important to provide mentoring-specific training that helps potential mentors improve their social skills (including coaching and other psychosocial support [7]), which can be decisive for the newcomers' onboarding success. P3 stated this as a problem with mentors: *"...the biggest challenges for mentor are: [...] engaging in other ways than just coding. These projects are about community effort and more than just the code."* The mentoring literature shows that a mentor can help shield a mentee from flaming wars with more senior members and intervene in certain situations to help them resolve it appropriately [42]. Thus, helping newcomers with interpersonal challenges and making them feel supported potentially helping newcomers getting comfortable and productive.

4.2 Supporting task recommendation

Finding an appropriate task for newcomers in OSS projects is challenging for mentors. This was also reported by Ann Barcomb and colleagues [8] in the context of episodic contributors. They found that community managers find it difficult to identify and maintain a list of tasks that can be picked up by newcomers to a project. In this paper, we evidenced that this process involves multiple challenges related to understanding newcomers' backgrounds and having a comprehensive knowledge about the tasks in the project. We also observed that the mentors have different strategies to overcome the challenges, which rarely rely on automated processes.

In particular, identifying task complexity is relevant since newcomers may feel demotivated if tasks are too simple or too complex. To determine task complexity, mentors reproduce the bugs, compare the bugs to other bugs on the issue tracker, and rely on tags, documentation, and discussion. Besides identifying task complexity, mentors determine the skills required for finishing the tasks. Although some mentors are aware of tools that can help identify the complexity of tasks and required skills, they do not find them useful for this purpose and instead rely on their judgment. When they are unable to find any tasks that would be appropriate for newcomers, mentors work to improve the project's task landscape by introducing additional functionality to the project or dividing tasks into smaller pieces.

In particular, mentors mentioned that they identify newcomers' characteristics by evaluating their interests and expertise, their confidence levels, and the skills they want to improve. Based on their mental models and the self-assessment of newcomers, mentors try to match newcomers and tasks. To support skill acquisition, mentors progressively increase the task complexity level to challenge newcomers and also assign repetitive tasks.

Concerning the strategies applied to identify and recommend the tasks to newcomer, we noticed that the findings of this study presents a considerable intersection with the practices proposed by Barcomb et al. [8]. For example, similarly to the strategy "Tagging the task based on difficulty," Barcomb et al. report that "Iden-

tify appropriate tasks" and "Define one-off tasks" are good practices for community preparation to episodic contributors.

Provide easily accessible information about tasks. When mentors assign tasks to newcomers, they need to filter them based on factors such as complexity and required skills. Without proper information, they must sift through the available tasks and read descriptions to triage tasks that are newcomer-friendly or "low-hanging fruits" [95]. Depending on the size and complexity of the project and the number of available tasks, this can be a complicated and tedious activity. For example, the project `kubernetes`[1] had more than 2,000 open tasks when this paper was written.

Labeling/tagging is a widespread practice for providing support in task selection. During this process, project members add labels to tasks, as described by our interviewees (Section 3.3). Our survey respondents acknowledged that this strategy aids in overcoming most of the challenges (see Table 4). Adding detail to a task can help both mentors and newcomers develop a more holistic picture of its complexity and solution, which facilitates the assessment of whether a task matches a newcomer profile. We suggest that tags inform about the tasks' appropriateness for newcomers in terms of required skills, priority, estimated effort, and difficulty. Although tagging is an approach recommended by GitHub and OSS project guidelines, sometimes projects do not have the capacity to triage and label tasks. This may lead to another problem: the lack of (explicitly) available tasks (Prod2.3). When a project adopts the labeling strategy, it is important to maintain and update the labeled issues to prevent newcomers from taking on already fixed or outdated issues [84]. Providing automated ways of tagging and better supporting human annotators are still open research opportunities that could provide great help to mentors and newcomers [69].

Explore different ways to understand newcomers' skills. Our results indicate that expertise and skill identification (Prod2.7) is a key challenge for mentors who recommend tasks to newcomers. The evidence collected here extends the understanding that awareness of developers' skills is the foundation for building productive teams [27, 78]. Baltes and Diehl [6] presented a theory mapping the main traits of software developers' expertise. Mentoring is explicitly presented as a central part of the theory, since it helps "building knowledge and thus contributes to the development of expertise." This theory also highlights that mentorship is a feedback mechanism that may help developers gain task-specific and general knowledge. Although the theory shines light on the importance of mentorship in knowledge acquisition, the authors of the theory could not find appropriate ways to objectively assess expertise. Our results echo this evidence, since we noticed that the mentors did not provide any objective way to assess newcomers' expertise.

In fact, mentors usually interact with newcomers to collect additional information to match their current skill level with the skills required for a specific project or task. This unstructured communication creates a strong mentor-mentee bond and facilitates the formation of ties that may influence future engagement. This strategy is in line with the landscape feature of proactive assistance and mentoring culture stated by Dagenais et al. [21], which was considered by their study as the most influ-

[1] https://github.com/kubernetes/kubernetes/issues

ential in how pleasant and efficient the integration experience was for the newcomer, and was also mentioned by seven participants as an effective way to reduce social fear (Prod.1). However, sometimes the gathered information can be subjective and influenced by low or high levels of self-confidence. Their self-assessment may not be accurate [6], because "if a person's pattern of past performance was highly variable in relation to relatively constant evaluation criteria [they] would not be able to form a stable assessment of his ability ... highly variable tasks may not be characterized by a person as skill task at all but as involving factors beyond his control" [28].

The strategy of directly asking newcomers to identify their skills may also be impacted by communication issues. The literature points out that professional developers experience gaps in communication (both written and oral communication) [62] and negotiation skills [45]. Additionally, the completely remote mentoring (e-mentoring) approach usually employed in OSS, as opposed to face-to-face interactions, may reduce trust between the parties [86] and likewise decrease communication effectiveness [13]. Thus, in some cases it may not reduce or eradicate newcomers' social fears [89], especially in the initial (or introductory) phase of the joining process [55]. Miscommunication on text-based channels is common, since context and expression is often lost [86]. However, as the literature indicates, e-mentoring—a computer-mediated alternative to the classic face-to-face mentoring—can scale up mentoring, providing more information to mentees and enabling them to connect with more people than classic face-to-face mentoring would allow [85]. E-mentoring has a particular applicability for OSS communities, as the work is conducted in a distributed way [90].

As several of our interviewees reported, another way to identify existing skills is by relying on a portfolio built on data from previous interactions with repositories and technical communities. Indeed, software development leaves traces of development activities in the repository, which can then be used for inferring the expertise of developers [20]. Existing tools such as "My GitHub Resume"[2] and "Visual Resume" [68] may help to assess developers' skills based on real interactions. However, these tools are not widely-used by OSS developers. Moreover, sometimes no historic data of a newcomer's contributions is available to evaluate expertise. Therefore, mentors who are unable to promptly identify newcomers' skills, can instead evaluate them by giving small tasks to the newcomers. This strategy allows them to explore the project while they have the chance to evaluate the newcomer's performance. This manner of assessing skills may be effective for contextualizing the newcomer's background within the project boundaries. However, it may create further issues if the tasks used for this evaluation are too easy or too complex, including demotivating newcomers and reducing retention. Therefore, mentors should explore multiple ways to understand newcomers' skills.

Create a pathway for newcomer learning and retention. According to our survey respondents, starting with smaller tasks and following them with more complex tasks is an effective strategy. Creating an adequate path and providing appropriate encouragement and support may also help in engaging and retaining newcomers. Creating these pathways depends not only on the skills of the newcomers, but on aspects such as developers' goals, motivations, and availability [69]. One op-

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

tion is to provide tasks that require higher competence of a particular skill and then move on to more complex and broad tasks. Another option is to keep newcomers working on tasks that fit their current skill set. The choice for the most appropriate alternative should align with the newcomers' goals, which requires constant follow-up from the mentor. The theory of Legitimate Peripheral Participation (LPP) has been widely adopted and describes how participation, situated learning, and identity construction are interrelated, and can evolve as a person joins a community of practice [36]. Instead of processing information in an isolated style, situated learning occurs though social interactions and puts a strong focus on practicing the acquired knowledge [23]. Fang et al. [26] suggested that OSS developers' learning behavior is situated in their everyday activities and that newcomer retention can occur after having repeated positive situated learning and identity construction social interactions. The strategy mentioned by our participants of creating an appropriate and evolutionary pathway of tasks with appropriate encouragement and support is also aligned with the situated learning of identity construction LPP.

4.3 Implications

Education and Training Personnel: People interested in Education and Training can make use of our findings to better understand the challenges faced by mentors in OSS. When asked whether they had been trained to act as a mentor, all of our participants answered "no." Therefore, it is important to offer training on the skills needed to be a mentor, either in undergraduate level or even in a professional environment. Moreover, given the number of social challenges revealed by the participants, it is important that (future) professionals acquire the proper "soft" skills that will better prepare them to mentoring. The challenges evidenced here also serve as a starting point for making instructors aware of what to expect when incorporating OSS projects in their curricula, which is becoming more common [12, 51, 60].

Online communities: Given the number of challenges mentors face (25), it is important that communities provide adequate support to those who volunteer or act as mentors. We found that providing up-to-date and straightforward documentation of available open issues and precise tagging of available open issues are some techniques that OSS communities can utilize to support both newcomers and mentors. We also identified a set of task recommendation strategies that can be used by mentors to recommend tasks to newcomers. Tagging the tasks (and keeping them up-to-date) was frequently recommended. In fact, this strategy is already in place in several well-established projects, like LibreOffice, Apache Open Office, Mozilla, Gnome, Media Wiki, and Ubuntu. Moreover, we found that "difference in the devices that mentors and mentees use" is also a challenge. Thus, providing ways to make it easier to build the system locally is of great benefit to the onboarding process. A potential solution would be a pre-configured environment, by means of a Virtual Machine with a built environment [95] or a container management tool, such as Docker.[3]

Mentors: Mentors can benefit from the set of challenges uncovered in this paper becoming more aware of what they can expect when dealing with newcomers, and better prepare themselves for supporting those willing to contribute to or join the

[3] http://www.docker.com

community. Mentors can also take advantage of the strategies presented in Section 3.3, employing them to support newcomers.

Researchers. This research described several challenges and strategies that could be further investigated and supported. Research is needed to help the community develop a more precise description of newcomer-friendly tasks. Observational studies may be conducted to understand what information newcomers look for when selecting a task and to provide insights about the dimensions that should become labels. Information needs and mentoring strategies for newcomers with different learning styles [14] could also be investigated. In addition, exploring how the newcomers actually find information may inform machine learning approaches to automatically suggesting labels for issues. Moreover, identifying better ways to elicit newcomers' characteristics, evaluate their effectiveness, and propose novel and more effective methods of evaluating newcomers' characteristics can be a potential research topic. Furthermore, future work may focus on investigating how each challenge and strategy influences newcomers with different motivations [33], retention in the project, and assertiveness in recommended tasks. Understanding motivation and demotivation factors influencing mentors can also be investigated in future research.

5 Limitations and Threats to Validity

Sampling bias could have affected our survey. However, the survey was widely deployed, receiving 600+ respondents who represented a wide set of demographics and projects. Nevertheless, although it mirrors the distribution of our population, we received very few answers from women, which may have contributed for the lack of statistically significant results when comparing men and women.

Sampling bias can also affect our interviews, including self-selection and social desirability biases. However, we counteracted this effect by seeking out different perspectives, inviting people with different profiles and diverse backgrounds and from various projects. We also acknowledge that the size of our sample (18 interviews) can be considered small. However, we continued interviewing until we were not able to identify any new challenges or strategies in two interviews in a row. The amount of participants is in line with the anthropology literature, which mentions that a minimum of 10 knowledgeable people is enough to uncover and understand the core categories in a study of lived experience [11]. Finding participants for this study was challenging because mentoring in such non-structured environments can take place through private, not publicly visible communication channels [24]. To increase the number of respondents in our study, we deployed multiple tactics to reach mentors, contact survey respondents, as well as reaching out to personal contacts (and snowballing) and previous contacts with Google Summer of Code mentors, social media posts, and OSS-related mailing lists. We relied on self-reported experience in mentoring to select our participants. During the analysis of the interviews we looked for evidence of their mentoring experience and we could not find any case for which we identified a mentor with low experience.

Another threat to validity can arise in the qualitative analysis process. However, to avoid misinterpretation in the qualitative coding of the data, we used the constant comparison method. As new codes emerged, we compared it with the existing code set and met frequently with the research team to discuss and clarify the codes. To increase construct validity, we worked closely with the D&I committee at the

ASF to craft the survey and interview questions such that it was accessible to the community.

However, we likely did not discover all possible challenges and strategies or provide complete explanation of them. We are aware that the OSS universe is vast, and challenges and strategies can differ according to projects.

Finally, this research focused on OSS settings to gain a deeper understanding of this specific community. Challenges and strategies may be different for companies, other online communities, and different types of users. Future research should focus on analyzing the commonalities and differences among challenges and strategies in different domains to build generalized models and theories about onboarding and mentorship in open collaboration communities.

6 Related work

Newcomers joining OSS projects face many challenges [81], and task assignment is a recurrent hurdle [80]. In the following, we provide more details about the existing literature related to mentorship in OSS projects, choosing appropriate tasks for newcomers, and strategies to support task recommendation.

6.1 Mentoring

Mentoring is explored in several domains and activities: in management literature is a way to help new employee socialization [1, 58, 88] (e.g., helping newcomers understanding the company's processes, the internal culture), and in education (teaching) literature is a way to help new teachers acclimate [48, 64, 65] and students overcome learning challenges [18, 34, 52]. For our purposes, a mentor is someone of advanced rank or experience who guides, teaches, and develops a novice [53]. Some existing literature analyzes the challenges faced during mentorship. For example, [63] conducts a literature review analyzing the challenges related to gender in the mentor-mentee relationship. In the education domain, [48] explores the problems encountered in mentoring new teachers, while [43] explore the challenges faced by faculty members while mentoring online doctoral students.

Software Engineering has also taken up mentoring as an object of study [9, 10, 75]. In fact, the importance of mentorship as part of the knowledge acquisition process for novices is evidenced in the theory of software development expertise developed by Baltes and Diehl [6]. In closed source settings, formal mentorship is a common practice to support newcomers [9]. Dagenais et al. [21] reported teams that proactively mentor newcomers make integration easier.

Formal mentorship in OSS has also been investigated. Some researchers focused on automatically identifying and recommending mentors [15, 47, 56, 77], claiming that mentoring benefits newcomers' onboarding. Fagerholm et al. [24] conducted a case study to assess the impact of mentoring and found that it significantly improves newcomer onboarding. In addition, Schilling et al. [70] studied the impact of mentoring on developers' training and retention and introduced a measure to assess mentoring capacity to facilitate learning and retention. In contrast, Labuschagne and Holmes [44], who studied Mozilla, evidenced that onboarding programs may not result in long-term contributors, even though mentored newcomers considered the mentorship program valuable. Silva et al. [74] investigated the strategies OSS projects use to mentor and onboard newcomers in the context of Summer of Code

programs. Mentorship is adopted by several prominent OSS projects (e.g., RedHat,[4] KDE,[5] Apache,[6] and OpenStack [7]).

Although the literature shows some interest in OSS mentorship and approaches to support finding an appropriate task for newcomers, to the best of our knowledge there is no systematic identification of strategies that mentors employ to help newcomers select their tasks and the challenges they face in this process. The present work adds to the OSS onboarding and task recommendation literature, as well as to the scarce literature about mentoring in open collaboration environments.

6.2 Task recommendation

The OSS literature widely reports the dilemma of finding an appropriate task, because new developers find it challenging to identify bugs that interest them, match their skill sets, are not duplicates, and are important for the community [94] [80]. For example, Park and Jensen [57] reported that newcomers need specific guidance on what to contribute (e.g., open issues, required features, and simple tasks to start with), while von Krogh et al. [92] reported that communities expect new participants to find tasks to work on, although they sometimes assign tasks. The literature also shows that newcomers struggle to find a task [80] while they often also face an arduous learning curve to handle the technical complexity, given the lack of domain knowledge or project information available for starters.

Some initiatives aim to support newcomers to find appropriate tasks, like Up For Grabs,[8] First Timers Only,[9] and Awesome for Beginners,[10] which aim to aggregate easy issues from several OSS projects. GitHub[11] encourages projects to tag issues that are easy for newcomers, which is a strategy also used by communities such as LibreOffice,[12] KDE,[13] and Mozilla. [14]

The literature also includes findings related to recommending and filtering out tasks. For instance, Cubranic et al. [19] presented Hipikat, a tool that builds a group memory and recommends source code, email messages, and bug reports to newcomers. Similarly, Wang and Sarma [94] previously developed a Tesseract extension that enables newcomers to identify similar bugs through synonym-based searches and to visually explore a bug's socio-technical dependencies.

Another way to support task recommendation is through systems that match people to tasks. Macdonald and Ounis [46] used a voting heuristic based on analyzing the modification history of each artifact related to the task. Anvik and Murphy [2] similarly use machine learning in the project's history to identify and suggest the developer most familiar with certain artifacts, identifying this person as an expert.

[4] https://wiki.gnome.org/Newcomers/Mentors
[5] https://community.kde.org/Mentoring
[6] https://community.apache.org/mentoringprogramme.html
[7] https://wiki.openstack.org/wiki/Mentoring
[8] https://up-for-grabs.net
[9] https://www.firsttimersonly.com/
[10] https://github.com/MunGell/awesome-for-beginners
[11] https://help.github.com/articles/helping-new-contributors-find-your-project-with-labels/
[12] https://wiki.documentfoundation.org/Development/EasyHacks
[13] https://community.kde.org/KDE/Junior_Jobs
[14] https://wiki.mozilla.org/Good_first_bug

Finally, DebugAdvisor [3] also aims to recommend developers based on expertise on the source code related to the task.

Although the literature explores strategies for task recommendation in OSS projects, they rely on extensive manual work to tag the issue tracker, since issue trackers usually either do not consider newcomers' skills and interests, or only work with developers who have previous experience in the project. In this work, we extend the existing literature by uncovering the strategies mentors use to recommend appropriate tasks for newcomers.

7 Conclusion

OSS communities frequently rely on mentors to guide newcomers to become long-term, active contributors. According to our data, contributors are less likely to be a mentor when they volunteer (not compensated to contribute) and have less than 1 year of experience.

We also identified 25 challenges faced by mentors in OSS projects. Further, we had a deep investigation of one of those challenges - *Difficulty in identifying appropriate tasks for newcomers* (ID Pro2 from Table 4). We identified 7 challenges that mentors face when recommending tasks to newcomers. The identified challenges range from handling newcomers' low self-efficacy and understanding their background to dealing with poor information about the tasks available at hand. We also identified 13 approaches to recommending tasks for newcomers. These strategies relate to identifying newcomers' characteristics, scaffolding newcomers' skill acquisition, identifying task complexity, identifying skills required to finish a task, and restructuring task landscape.

To conclude mentorship in OSS is still an under explored research area. Our results provide initial insights about characteristics of mentors and challenges they face and strategies they employ, which OSS communities and mentors can leverage to improve the current state of practice. Providing better support for mentorship can help to attract more volunteers to this important role and, indirectly, provide smoother onboarding of newcomers to OSS projects.

Abbreviations

- ASF: Apache Software Foundation
- LPP: Legitimate Peripheral Participation
- OSS: Open Source Software
- RQ1: Research Question 1
- RQ2: Research Question 2
- RQ3: Research Question 3
- VIF: Variable Inflation Factors

Availability of data and materials

The datasets generated and analysed during the current study are not publicly available due to the terms of the consent.

Competing interests

The authors declare that they have no competing interests.

Funding

This work is partially supported by the National Science Foundation under Grant Numbers 1815486, 1815503, 1900903, and 1901031.

Authors' contributions

IS, AS, and MAG took part in the whole process of both phases, working on the the design, data collection, analysis, discussion, supervision, and report of the study; SB was responsible for Stage 2 collection, analysis and report; BT worked on data analysis and report. MG, DIC and GGCZ worked on design, data collection and analysis, and report of Stage 1. All authors read and approved the final manuscript.

Acknowledgements

We thank all interviewees and survey participants for their contribution to this research. We are also grateful to the ASF for their support.

Author details

[1]Department of Computing, Universidade Tecnológica Federal do Paraná, Campo Mourao, PR, Brazil. [2]School of Electrical Engineering and Computer Science, Oregon State University, Corvalis, OR, USA. [3]School of Informatics, Computing & Cyber Systems, Northern Arizona University, Flagstaff, AZ, USA. [4], Bitergia, Madrid, Spain. [5], Apache Software Foundation, Wakefield, MA, USA.

References

1. Tammy D Allen, Lillian T Eby, Georgia T Chao, and Talya N Bauer. Taking stock of two relational aspects of organizational life: Tracing the history and shaping the future of socialization and mentoring research. *Journal of Applied Psychology*, 102(3):324–337, March 2017. ISSN 1939-1854 (Electronic); 0021-9010 (Print). . URL http://dx.doi.org/10.1037/apl0000086.
2. John Anvik and Gail C. Murphy. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. *ACM Trans. Softw. Eng. Methodol.*, 20(3):10:1–10:35, August 2011. ISSN 1049-331X. . URL http://doi.acm.org/10.1145/2000791.2000794.
3. B. Ashok, Joseph Joy, Hongkang Liang, Sriram K. Rajamani, Gopal Srinivasa, and Vipindeep Vangala. Debugadvisor: A recommender system for debugging. In *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ESEC/FSE '09, pages 373–382, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-001-2. . URL http://doi.acm.org/10.1145/1595696.1595766.
4. Sogol Balali, Igor Steinmacher, Umayal Annamalai, Anita Sarma, and Marco Aurelio Gerosa. Newcomers' barriers. . . is that all? an analysis of mentors' and newcomers' barriers in oss projects. *Computer Supported Cooperative Work (CSCW)*, 27(3):679–714, Dec 2018. ISSN 1573-7551. . URL https://doi.org/10.1007/s10606-018-9310-8.
5. Sogol Balali, Umayal Annamalai, Hema Susmita Padala, Bianca Trinkenreich, Marco A Gerosa, Igor Steinmacher, and Anita Sarma. Recommending tasks to newcomers in oss projects: How do mentors handle it? In *Proceedings of the 16th International Symposium on Open Collaboration*, pages 1–14, 2020.
6. Sebastian Baltes and Stephan Diehl. Towards a theory of software development expertise. In *ACM Symposium on the Foundations of oftware Engineering (FSE 2018)*, pages 187–200, 2018.
7. Lisa E Baranik, Elizabeth A Roling, and Lillian T Eby. Why does mentoring work? the role of perceived organizational support. *Journal of vocational behavior*, 76(3):366–373, 2010. . URL https://doi.org/10.1016/j.jvb.2009.07.004.
8. Ann Barcomb, Klaas-Jan Stol, Brian Fitzgerald, and Dirk Riehle. Managing episodic volunteers in free/libre/open source software communities. *IEEE Transactions on Software Engineering*, 2020.
9. Andrew Begel and Beth Simon. Novice software developers, all over again. In *Proceedings of the Fourth International Workshop on Computing Education Research*, ICER '08, pages 3–14, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-216-0. . URL http://doi.acm.org/10.1145/1404520.1404522.
10. Lucy M. Berlin. Beyond program understanding: A look at programming expertise in industry. Technical Report HPL-92-142, Hewlett-Packard Laboratories, Palo Alto, CA, USA, 1992. http://www.hpl.hp.com/techreports/92/HPL-92-142.html. Accessed in 18 February 2018.
11. H Russell Bernard. *Research methods in anthropology: Qualitative and quantitative approaches*. Rowman & Littlefield, 2017.
12. Judith Bishop, Carlos Jensen, Walt Scacchi, and Arfon Smith. How to use open source software in education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 321–322, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3685-7. . URL http://doi.acm.org/10.1145/2839509.2844665.
13. Kevin Buffardi. Comparing remote and co-located interaction in free and open source software engineering projects. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 22–27, 2017.
14. Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2586–2598, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. . URL http://doi.acm.org/10.1145/2858036.2858274.
15. Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. Who is going to mentor newcomers in open source projects? In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, pages 44:1–44:11, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1614-9. . URL http://doi.acm.org/10.1145/2393596.2393647.
16. Tadeusz Chełkowski, Peter Gloor, and Dariusz Jemielniak. Inequalities in open source software development: Analysis of contributor's commits in apache software foundation projects. *PLoS One*, 11(4):e0152976, 2016.
17. Boyuan Chen and Zhen Ming Jack Jiang. Characterizing logging practices in java-based open source software projects–a replication study in apache software foundation. *Empirical Software Engineering*, 22(1):330–374, 2017.
18. Gloria Crisp and Irene Cruz. Mentoring college students: A critical review of the literature between 1990 and 2007. *Research in higher education*, 50(6):525–545, 2009. ISSN 1573-188X. . URL https://doi.org/10.1007/s11162-009-9130-2.
19. Davor Cubranic, Gail C. Murphy, Janice Singer, and Kellogg S. Booth. Hipikat: a project memory for software development. *IEEE Transactions on Software Engineering*, 31(6):446–465, June 2005.
20. Jose Ricardo da Silva, Esteban Clua, Leonardo Murta, and Anita Sarma. Niche vs. breadth: Calculating expertise over time through a fine-grained analysis. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 409–418. IEEE, 2015.

21. Barthélémy Dagenais, Harold Ossher, Rachel K. E. Bellamy, Martin P. Robillard, and Jacqueline P. de Vries. Moving into a new software project landscape. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 275–284, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-719-6. . URL http://doi.acm.org/10.1145/1806799.1806842.
22. David L DuBois, Bruce E Holloway, Jeffrey C Valentine, and Harris Cooper. Effectiveness of mentoring programs for youth: A meta-analytic review. *American journal of community psychology*, 30(2):157–197, 2002. . URL https://link.springer.com/article/10.1023/A:1014628810714.
23. Amy Edmondson. Psychological safety and learning behavior in work teams amy edmondson. *Administrative Science Quarterly*, 44(2):350–383, 1999.
24. Fabian Fagerholm, Alejandro S. Guinea, Jürgen Münch, and Jay Borenstein. The role of mentoring and project characteristics for onboarding in open source software projects. In *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '14, pages 55:1–55:10. ACM, 2014. ISBN 978-1-4503-2774-9. . URL http://doi.acm.org/10.1145/2652524.2652540.
25. Fabian Fagerholm, Alejandro S Guinea, Jürgen Münch, and Jay Borenstein. The role of mentoring and project characteristics for onboarding in open source software projects. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–10, 2014.
26. Yulin Fang and Derrick Neufeld. Understanding sustained participation in open source software projects. *Journal of Management Information Systems*, 25(4):9–50, April 2009. ISSN 0742-1222. . URL http://dx.doi.org/10.2753/MIS0742-1222250401.
27. Samer Faraj and Lee Sproull. Coordinating expertise in software development teams. *Management science*, 46 (12):1554–1568, 2000.
28. N. T. Feather. Attribution of responsibility and valence of success and failure in relation to initial confidence and task performance. *Journal of Personality and Social Psychology*, 13(2), 1969. ISSN 1939-1315 (Electronic); 0022-3514 (Print). . URL http://psycnet.apa.org/fulltext/1970-00602-001.pdf.
29. Jane Forman and Laura Damschroder. Qualitative content analysis. *Empirical methods for bioethics: A primer*, 11:39–62, 2007.
30. Andrea Forte and Cliff Lampe. Defining, understanding, and supporting open collaboration: Lessons from the literature. *American Behavioral Scientist*, 57(5):535–547, 2013. . URL https://doi.org/10.1177/0002764212469362.
31. John Fox. *Applied regression analysis and generalized linear models*. Sage Publications, 2015.
32. D Randy Garrison, Martha Cleveland-Innes, Marguerite Koole, and James Kappelman. Revisiting methodological issues in transcript analysis: Negotiated coding and reliability. *The Internet and Higher Education*, 9(1):1–8, 2006.
33. Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. The shifting sands of motivation: Revisiting what drives contributors in open source. In *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*, 2021.
34. Susan Gershenfeld. A review of undergraduate mentoring programs. *Review of Educational Research*, 84(3): 365–391, 2014. . URL https://doi.org/10.3102/0034654313520512.
35. Mohammad Gharehyazie, Daryl Posnett, Bogdan Vasilescu, and Vladimir Filkov. Developer initiation and social interactions in oss: A case study of the apache software foundation. *Empirical Software Engineering*, 20(5): 1318–1353, 2015.
36. Karen Handley, Andrew Sturdy, Robin Fincham, and Timothy Clark. Within and beyond communities of practice: Making sense of learning through participation, identity and practice. *Journal of management studies*, 43(3):641–653, 2006.
37. Erin Hatton. Mechanisms of invisibility: rethinking the concept of invisible work. *Work, Employment and Society*, 31(2):336–351, 2017. .
38. Gary Hsieh, Youyang Hou, Ian Chen, and Khai N. Truong. "welcome!": Social and psychological predictors of volunteer socializers in online communities. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 827–838, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1331-5. . URL http://doi.acm.org/10.1145/2441776.2441870.
39. P. N. Johnson-Laird. *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Harvard University Press, Cambridge, MA, USA, 1983. ISBN 0-674-56882-6.
40. Philip N. Johnson-Laird. Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250, 2010. ISSN 0027-8424. . URL http://www.pnas.org/content/107/43/18243.
41. Suhas Kabinna, Cor-Paul Bezemer, Weiyi Shang, and Ahmed E Hassan. Logging library migrations: A case study for the apache software foundation projects. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 154–164. IEEE, 2016.
42. Kathy E Kram. *Mentoring at work: Developmental relationships in organizational life*. University Press of America, 1988. ISBN 0-8191-6755-X. URL http://psycnet.apa.org/record/1988-97625-000.
43. Swapna Kumar, Melissa Johnson, and Truly Hardemon. Dissertations at a distance: Students perceptions of online mentoring in a doctoral program. *International Journal of E-Learning & Distance Education*, 27(1), 2013. URL http://www.ijede.ca/index.php/jde/article/view/835.
44. Adriaan Labuschagne and Reid Holmes. Do onboarding programs work? In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 381–385, Piscataway, NJ, USA, 2015. IEEE Press. ISBN 978-0-7695-5594-2. . URL http://dl.acm.org/citation.cfm?id=2820518.2820567.
45. Timothy C Lethbridge. What knowledge is important to a software professional? *Computer*, 33(5):44–50, 2000.
46. Craig Macdonald and Iadh Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396. ACM, 2006.
47. Yuri Malheiros, Alan Moraes, Cleyton Trindade, and Silvio Meira. A source code recommender system to support newcomers. In *Proceedings of the IEEE 36th Annual Computer Software and Applications Conference*, COMPSAC '12, pages 19–24, Los Alamitos, California, USA, 2012. IEEE. . URL

http://ieeexplore.ieee.org/document/6340250/.
48. Kay Martinez. Mentoring new teachers: Promise and problems in times of teacher shortage. *Australian Journal of Education*, 48(1):95–108, 2004. . URL https://doi.org/10.1177/000494410404800107.
49. Ann Mihkelson. A model of research mentoring for higher education–an overview. Technical report, University of Tasmania (Australia), 1997.
50. David R. Musicant, Yuqing Ren, James A. Johnson, and John Riedl. Mentoring in wikipedia: A clash of cultures. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, WikiSym '11, pages 173–182, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0909-7. . URL http://doi.acm.org/10.1145/2038558.2038586.
51. Debora Nascimento, Kenia Cox, Thiago Almeida, Wendell Sampaio, Roberto Bittencourt, Rodrigo Souza, and Christina Chavez. Using open source projects in software engineering education: A systematic mapping study. In *IEEE Frontiers in Education Conference*, pages 1837–1843. IEEE, 2013. .
52. Katherine E Nugent, Gwen Childs, Rosalind Jones, and Pamela Cook. A mentorship model for the retention of minority students. *Nursing Outlook*, 52(2):89–94, 2004. URL https://doi.org/10.1016/j.outlook.2003.09.008.
53. Gordon O'Brien. Methods of analyzing group tasks. Technical report, Department of Psychology, University of Illinois, Urbana, URBANA, ILLINOIS , USA, 1967.
54. Gustavo Ansaldi Oliva, José Teodoro da Silva, Marco Aurélio Gerosa, Francisco Werther Silva Santana, Cláudia Maria Lima Werner, Cleidson Ronald Botelho de Souza, and Kleverton Carlos Macedo de Oliveira. Evolving the system's core: a case study on the identification and characterization of key developers in apache ant. *Computing and Informatics*, 34(3):678–724, 2015.
55. Ilan Oshri, Julia Kotlarsky, and Leslie P Willcocks. Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects. *The Journal of Strategic Information Systems*, 16(1): 25–49, 2007.
56. Sebastiano Panichella. Supporting newcomers in software development projects. In *IEEE International Conference on Software Maintenance and Evolution*, ICSME 2015, pages 586–589. IEEE, 2015. . URL http://ieeexplore.ieee.org/document/7332519/.
57. Yunrim Park and Carlos Jensen. Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In *2009 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 3–10. IEEE, 2009.
58. Stephanie C Payne and Ann H Huffman. A longitudinal examination of the influence of mentoring on organizational commitment and turnover. *Academy of Management Journal*, 48(1):158–168, February 2005. .
59. Gustavo Pinto, Igor Steinmacher, and Marco Gerosa. More common than you think: An in-depth study of casual contributors. In *SANER*, pages 112–123, 2016.
60. Gustavo Pinto, Igor Steinmacher, Fernando Figueira Filho, and Marco A. Gerosa. Training the next generation of software engineers using open-source software: An interview study. In *IEEE 30th International Conference on Software Engineering Education and Training*, CSEET 2017, Los Alamitos, California, USA, 2017. IEEE.
61. Teade Punter, Marcus Ciolkowski, Bernd Freimut, and Isabel John. Conducting on-line surveys in software engineering. In *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.*, pages 80–88. IEEE, 2003.
62. Alex Radermacher and Gursimran Walia. Gaps between industry expectations and the abilities of graduates. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 525–530, 2013.
63. Belle Rose Ragins. Barriers to mentoring: The female manager's dilemma. *Human Relations*, 42(1):1–22, January 1989. . URL http://dx.doi.org/10.1177/001872678904200101.
64. Donna Redman, Sharon Conley, and Terrence E. Deal. A cultural approach to mentoring new teachers. In Bruce S. Cooper and Carlos R. McCray, editors, *Mentoring for school quality: How educators can be more professional and effective*, pages 65–80. Rowman & Littlefield, Lanham, Maryland, 2015.
65. Jonah E Rockoff. Does mentoring reduce turnover and improve skills of new employees? evidence from teachers in new york city. Technical report, National Bureau of Economic Research, 2008.
66. Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131, 2009. . URL http://dx.doi.org/10.1007/s10664-008-9102-8.
67. Neil J Salkind. *Encyclopedia of educational psychology*. Sage Publications, 2008.
68. Anita Sarma, Xiaofan Chen, Sandeep Kuttal, Laura Dabbish, and Zhendong Wang. Hiring in the global stage: Profiles of online contributions. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, pages 1–10, Aug 2016. .
69. Anita Sarma, Marco Aurélio Gerosa, Igor Steinmacher, and Rafael Leano. Training the future workforce through task curation in an oss ecosystem. In *24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 932–935, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4218-6. . URL http://doi.acm.org/10.1145/2950290.2983984.
70. Andreas Schilling, Sven Laumer, and Tim Weitzel. Who will remain? an evaluation of actual person-job and person-team fit to predict developer retention in floss projects. In *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*, HICSS '12, pages 3446–3455. IEEE Computer Society, 2012. . URL http://ieeexplore.ieee.org/document/6149241/.
71. Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, July 1999. ISSN 0098-5589. . URL http://ieeexplore.ieee.org/document/799955/.
72. F. Sharan. Asf committer diversity survey - available at: https://cwiki.apache.org/confluence/display/comdev/asf+committer+diversity+survey+-+2016. accessed: 2020-10-16, 2016.
73. Jefferson Silva, Igor Scaliante Wiese, Daniel German, Igor Steinmacher, and Marco Gerosa. How long and how much: What to expect from summer of code participants? In *IEEE International Conference on Software Maintenance and Evolution*, ICSME2017, pages 67–69, Los Alamitos, California, USA, 2017. IEEE. . URL

http://ieeexplore.ieee.org/document/8094410/.
74. Jefferson Silva, Igor Wiese, Daniel German, Christoph Treude, Marco Gerosa, and Igor Steinmacher. A theory of the engagement in open source projects via summer of code programs. In *Proceedings of the ACM Symposium on the Foundations of Software Engineering (FSE 2020)*, FSE '20, 2020.
75. Susan E. Sim and Richard C. Holt. The ramp-up problem in software projects: a case study of how software immigrants naturalize. In *Proceedings of the 20th International Conference on Software Engineering*, ICSE '98, pages 361–370. IEEE, April 1998. . URL http://ieeexplore.ieee.org/document/671389/.
76. Donna Spencer. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.
77. Igor Steinmacher, Igor Scaliante Wiese, and Marco Aurélio Gerosa. Recommending mentors to software project newcomers. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, RSSE '12, pages 63–67, Washington, DC, USA, June 2012. IEEE Computer Society. . URL http://ieeexplore.ieee.org/document/6233413/.
78. Igor Steinmacher, Ana Paula Chaves, and Marco Aurélio Gerosa. Awareness support in distributed software development: A systematic review and mapping of the literature. *Computer Supported Cooperative Work (CSCW)*, 22(2-3):113–158, 2013. ISSN 0925-9724. . URL http://dx.doi.org/10.1007/s10606-012-9164-4.
79. Igor Steinmacher, Marco Aurélio Gerosa, and David Redmiles. Attracting, onboarding, and retaining newcomer developers in open source software projects. In *Proceedings of the Workshop on Global Software Development in a CSCW Perspective*, CSCW '14 Workshops, 2014.
80. Igor Steinmacher, Tayana Conte, and Marco Aurélio Gerosa. Understanding and supporting the choice of an appropriate task to start with in open source software communities. In *2015 48th Hawaii International Conference on System Sciences*, HICSS '15, pages 5299–5308. IEEE, 2015.
81. Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David F. Redmiles. Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 1379–1392, New York, NY, USA, 2015. ACM. URL http://doi.acm.org/10.1145/2675133.2675215.
82. Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. Overcoming open source project entry barriers with a portal for newcomers. In *38th International Conference on Software Engineering*, ICSE '16, pages 273–284, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3900-1. . URL https://dl.acm.org/citation.cfm?id=2884806.
83. Igor Steinmacher, Gustavo Pinto, Igor Wiese, and Marco A. Gerosa. Almost there: A study on quasi-contributors in open-source software projects. In *40th International Conference on Software Engineering*, ICSE'18, page 12, New York, NY, USA, 2018. ACM.
84. Igor Steinmacher, Christoph Treude, and Marco Gerosa. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software*, pages 1–1, 2018. ISSN 0740-7459. .
85. Heidrun Stoeger, Xiaoju Duan, Sigrun Schirner, Teresa Greindl, and Albert Ziegler. The effectiveness of a one-year online mentoring program for girls in stem. *Computers & Education*, 69:408–418, 2013.
86. Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M German. How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering*, 43(2):185–204, 2016.
87. Anselm Strauss and Juliet M. Corbin. *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 3rd edition, 2007. ISBN 0803959400.
88. Chris Street. Examining learning to teach through a social lens: How mentors guide newcomers into a professional commuity of learners. *Teacher Education Quarterly*, 31(2):7–24, 2004.
89. Gil Taran and Lynn Carter. Improving distance mentoring: Challenges and how to deal with them in global development project courses. In *Conference on Software Engineering Education and Training (CSEET)*, pages 97–104. IEEE, 2010.
90. Erik H Trainer, Arun Kalyanasundaram, and James D Herbsleb. E-mentoring for software engineering: a socio-technical perspective. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 107–116. IEEE, 2017.
91. Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G. J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. Gender and tenure diversity in GitHub teams. In *CHI Conference on Human Factors in Computing Systems*, CHI'15, pages 3789–3798, New York, NY, USA, 2015. ACM. . URL https://dl.acm.org/citation.cfm?id=2702549.
92. Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, 32(7):1217–1241, 2003.
93. Georg Von Krogh, Stefan Haefliger, Sebastian Spaeth, and Martin W Wallin. Carrots and rainbows: Motivation and social practice in open source software development. *MIS quarterly*, pages 649–676, 2012.
94. Jianguo Wang and Anita Sarma. Which bug should i fix: helping new developers onboard a new project. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '11, pages 76–79, New York, NY, USA, 2011. ACM.
95. Vincent Wolff-Marting, Christoph Hannebauer, and Volker Gruhn. Patterns for tearing down contribution barriers to floss projects. In *Proceedings of the 12th International Conference on Intelligent Software Methodologies, Tools and Techniques*, SoMeT '13, pages 9–14. IEEE, 2013. . URL http://ieeexplore.ieee.org/document/6645669/.