Recommending Tasks to Newcomers in OSS Projects: How Do Mentors Handle It?

Sogol Balali
Oregon State University
balalis@oregonstate.edu

Umayal Annamalai
Oregon State University
annamalu@oregonstate.edu

Hema Susmita Padala
Oregon State University
padalah@oregonstate.edu

Bianca Trinkenreich
Northern Arizona University
bt473@nau.edu

Marco A. Gerosa
Northern Arizona University
marco.gerosa@nau.edu

Igor Steinmacher
Northern Arizona University
igor.steinmacher@nau.edu

Anita Sarma
Oregon State University
anita.sarma@oregonstate.edu

ABSTRACT

Software developers who want to start contributing to an Open Source Software (OSS) project often struggle to find appropriate first tasks. The voluntary, self-organizing distribution of decentralized labor and the distinct nature of some OSS projects intensifies this challenge. Mentors, who work closely with newcomers, develop strategies to recommend tasks. However, to date neither the challenges mentors face in recommending tasks nor their strategies have been formally documented or studied. In this paper, we interviewed mentors of well-established OSS projects (n=10) and qualitatively analyzed their answers to identify both challenges and strategies related to recommending tasks for newcomers. Then, we employed a survey (n=30) to map the strategies to challenges and collect additional strategies. Our study identified 7 challenges and 13 strategies related to task recommendation. Strategies such as "tagging the issues based on difficulty," "adding documentation," "assigning a small task first and then challenge the newcomers with bigger tasks," and "dividing tasks into smaller pieces" were frequently mentioned as ways to overcome multiple challenges. Our results provide insights for mentors about the strategies OSS communities can use to guide their mentors and for tool builders who design automated support for task assignment.

CCS CONCEPTS

• **Software and its engineering → Open source model**; • **Human-centered computing → Open source software**; • **Social and professional topics** → *Project and people management*;

KEYWORDS

Task Recommendation, OSS, Mentoring, Newcomers

OpenSym 2020, August 25–27, 2020, Virtual conference, Spain

ACM ISBN 978-1-4503-8779-8/20/08…$15.00
https://doi.org/10.1145/3412569.3412571

ACM Reference Format:
Sogol Balali, Umayal Annamalai, Hema Susmita Padala, Bianca Trinkenreich, Marco A. Gerosa, Igor Steinmacher, and Anita Sarma. 2020. Recommending Tasks to Newcomers in OSS Projects: How Do Mentors Handle It?. In *16th International Symposium on Open Collaboration (OpenSym 2020), August 25–27, 2020, Virtual conference, Spain.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3412569.3412571

1 INTRODUCTION

Open collaboration communities, such as many Open Source Software (OSS) projects, aim to provide environments with low barriers to entry in order to onboard and retain newcomers [26]. However, previous work shows that newcomers to OSS communities face a variety of barriers that hinder their onboarding [62, 64]. One of the most common and challenging barriers is choosing an appropriate task [61]. Selecting an appropriate task is difficult and can be a time-consuming and frustrating activity; ultimately, this barrier to onboarding may lead to dropouts [61, 72].

The literature discusses some solutions to support task selection, such as automated recommendation systems [2, 16, 35, 73]. However, these systems only suggest tasks to developers with previous interactions in the project, and thus offer limited support for newcomers (this is known as the "cold start" problem). Moreover, the number of open tasks in popular projects' issue trackers, which developers need to skim to find a good match, can reach hundreds or a couple thousands (e.g., tensorflow,[1] react-native,[2] LibreOffice Writer[3]). As there is no systematic identification of strategies to support task recommendation in OSS, project members and newcomers need to develop their own strategies by trial and error, leading to the aforementioned barriers [61].

Mentors can provide a valuable perspective since they work closely with several newcomers and over time develop strategies for recommending tasks [8, 20]. In fact, mentorship is adopted by several prominent OSS projects (e.g., RedHat,[4] KDE,[5] Apache,[6] and OpenStack [7]). The literature shows that mentorship is beneficial in

[1] https://github.com/tensorflow/tensorflow/issues
[2] https://github.com/facebook/react-native/issues
[3] https://bugs.documentfoundation.org/buglist.cgi?component=Writer&product=LibreOffice&resolution=---
[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

both OSS [20, 21] and closed-source software development [8, 58] contexts. The literature also shows that OSS mentors are not usually formally trained and either voluntarily elect to mentor out of personal interest, or are asked to do so by the community [4]. Since they are untrained, even if they are technically resourceful mentors are not aware of the strategies that can be used to deal with task recommendation. No previous research has focused on the challenges mentors face when recommending tasks for newcomers nor on the strategies they adopt to recommend appropriate tasks for their mentees. Thus, scientifically identifying strategies that mentors adopt to support task recommendation can generate awareness among OSS communities, mentors, and researchers about the existing task recommendation challenges and possible strategies to overcome them.

Our goal is to identify strategies that OSS mentors adopt to cope with the challenges of recommending tasks to newcomers. To achieve this goal, we first interviewed OSS mentors to identify an initial list of strategies, and then conducted a follow-up survey to triangulate and complement the results obtained in the interviews. In the context of this research, a mentor is a peer who was assigned or volunteered to support newcomers who are onboarding to the project. Mentors are usually peers who succeeded in overcoming the project challenges and are willing to help others onboard [38].

In summary, this paper contributes to the literature by describing seven challenges faced by mentors when recommending tasks for newcomers, categorizing and detailing 13 strategies mentors employ to recommend tasks for newcomers, mapping mentors' strategies to challenges, and proposing guidelines to improve task recommendation for newcomers. Our results benefit newcomers who struggle with entry barriers associated with task selection, and mentors who aim to support them. Our results also inform the broader OSS community and tool builders by providing inputs that may be used to create mechanisms and processes to better support task recommendations.

2 RELATED WORK

Newcomers joining OSS projects face many barriers [62], and task assignment is a recurrent hurdle [61]. In the following, we provide more details about the existing literature related to choosing appropriate tasks for newcomers, strategies to support task recommendation, and mentorship in OSS projects.

2.1 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 [61, 73]. For example, Park and Jensen [45] 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. [72] 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 [45, 61] 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. [16] presented Hipikat, a tool that builds a group memory and recommends source code, email messages, and bug reports to newcomers. Similarly, Wang and Sarma [73] 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 [35] 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. 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.

2.2 Mentoring

Mentoring is explored in several domains and activities: in management literature is a way to help new employee socialization [1, 46, 69] (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 [37, 51, 52] and students overcome learning challenges [15, 28, 41]. For our purposes, a mentor is someone of advanced rank or experience who guides, teaches, and develops a novice [42]. Some existing literature analyzes the challenges faced during mentorship. For example, [50] conducts a literature review analyzing the challenges related to gender in the mentor-mentee relationship. In the education domain, [37] explores the problems encountered in mentoring new teachers, while [32] explore the challenges faced by faculty members while mentoring online doctoral students.

Software Engineering has also taken up mentoring as an object of study [8, 9, 58]. 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

[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

practice to support newcomers [8]. Dagenais et al. [18] reported teams that proactively mentor newcomers make integration easier.

Formal mentorship in OSS has also been investigated. Some researchers focus on automatically identifying and recommending mentors [14, 36, 44, 65], claiming that mentoring benefits newcomers' onboarding. Fagerholm et al. [20] conducted a case study to assess the impact of mentoring and found that it significantly improves newcomer onboarding. In addition, Schilling et al. [56] 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 [33], 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. [57] investigated the strategies OSS projects use to mentor and onboard newcomers in the context of Summer of Code programs. In previous work, Balali et al. [4] also analyzed challenges mentors face when onboarding newcomers.

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.

3 RESEARCH PLANNING AND EXECUTION

In this section, we present our study planning (Section 3.1), data collection (Section 3.2), and analysis (Section 3.3).

3.1 Study Planning

The goal of our study is to identify the challenges mentors face and the strategies they employ to recommend tasks to newcomers in OSS projects. To achieve this goal, we defined the following research questions (RQs):

> **RQ1.** What are the challenges that mentors face when recommending tasks to newcomers?
> **RQ2.** What are the strategies that mentors employ to recommend tasks?
> **RQ3.** Which strategies help mitigate challenges when recommending tasks to newcomers?

To answer our RQs, we designed a study combining interviews and survey. From the interviews, we derived a set of challenges and strategies. From the survey, we further investigated the mapping of strategies to challenges and collected additional strategies. Figure 1 depicts the overview of our research method.

Recruiting our participants was challenging since mentors are usually not listed in the project documentation and some projects do not formalize this role. Moreover, mentors are often core project members, performing mentoring activities alongside other technical ones. Therefore, to recruit the interview participants, we first contacted two mentors using a convenience sample, and then applied the snowballing strategy. After completing each interview, we asked the interviewee to recommend another qualified OSS mentor for us to contact. Then, we sent the candidate a recruitment e-mail, which asked them about their experience in onboarding and mentoring newcomers. We included in our sample mentors from well-established and largely used OSS projects with at least 2 years of mentoring experience.

In total, we contacted 18 mentors, of whom we interviewed ten. Out of these ten participants, two self-identified as women, and eight as men. All of our interviewees had a minimum of two years of experience mentoring newcomers in OSS communities. Additionally, five participants had industry experience, and one had experience working in academia. All of our interviewees received a 25-dollar gift card in compensation for their participation. Table 1 illustrates the demographic information of our study participants.

Table 1: Demographics for the Interview Participants

ID	Gender	Years of experience			OSS Project
		Mentor	OSS	Industry	
P1	M	8	5	11	Apache Lucene, Solar
P2	M	2	3	–	Gnome
P3	M	3	3	–	RedHat
P4	M	3	5	6	RedHat
P5	M	2.5	7	–	Gnome
P6	F	4	11	–	Linux Kernel, Apache Spark
P7	M	5	5	37	Not mentioned
P8	M	30	15	> 20	Linux kernel
P9	M	16	9	–	KDE
P10	F	4	5	0.5	Open Hatch

For the survey, we sent e-mails and posted advertisements on social networks and mailing lists. We received 31 answers, from which we discarded one due to incompleteness. Among our respondents, 24 self-identified as men and 6 as women. Their experience in open source averaged 6.8 years (min: 3, max: 15), and as a mentor 7.8 years (min: 2, max: 30).

3.2 Study Execution: Data Collection

Our first data set was collected through semi-structured interviews. The interviews were conducted around the central question: *"How do you recommend tasks to newcomers and what do you take into consideration?"*. We followed up by asking about specific challenges and scenarios in which the challenges occurred and how they acted in such cases. To validate our script and measure time, we conducted four pilot studies with Ph.D. students who had mentoring experience in industry or OSS environments. The results of these pilot interviews were discarded.

All interviews were remotely undertaken by two researchers using Skype, Google Hangouts, or phone, and lasted about 40 minutes. With participant consent, we audio-recorded all interviews, which were transcribed for the analysis. The interview script and the survey instrument are available as part of the supplementary material.[15]

The survey comprised three sections: demographics, mapping of strategies to challenges, and open questions. To map strategies to challenges, we presented a matrix, in which the mentors were asked to check which of the identified strategies can mitigate the challenges we identified. In the open questions, we asked about additional strategies (17 out of 31 answered this question); and more details about the successful application of any of the strategies (answered by 24 out of 31 participants).

[15] https://figshare.com/s/e4d1a8a13a9f5db9202d

3.3 Study Execution: Data Analysis

Each interview was manually transcribed, and then the first, second, and third authors followed a card sorting approach [59] 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 in a weekly basis with the other three more experienced authors. The whole process was conducted using continuous comparison [68] during the coding sessions and negotiated agreement [27] (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 [25, 27].

For the survey, we quantitatively analyzed the relationships between challenges and strategies, and conducted a process similar to the one used for the interviews to gain insights about the use of the strategies reported in the open-ended questions.

4 RESULTS

Our study revealed challenges faced (**RQ1**) and strategies employed (**RQ2**) by mentors when recommending tasks to newcomers. The codebook summarizing the categories found is available as a supplementary material.[16]

The answers to the first two research questions were input to **RQ3**, in which we focused on matching strategies and challenges faced while recommending tasks. In this section, we report our findings per research question, illustrated by quotes from the participants, evidence from the survey, and our observations.

4.1 What are the challenges that mentors face when recommending tasks to newcomers? (RQ1)

The analysis of the interviews resulted in seven categories of challenges faced by mentors while helping newcomers choose their tasks. These challenges are shown in Table 2 and discussed as follows.

Table 2: Challenges faced by mentors for recommending tasks for newcomers

Challenge name	ID
Challenging tasks can create social fears in the newcomers	C1
Mentors have to deal with newcomers' lack of holistic understanding about the project and its culture	C2
Lack of information about how newcomer-friendly a task is	C3
Difficulty in identifying the complexity of a task	C4
Difficulty in estimating the amount of time necessary to finish a task	C5
Lack of friendly tasks available for newcomers	C6
Lack of available information about newcomer's skills, interest, and expertise	C7

(C1) Challenging tasks can create social fears in the newcomers (mentioned by P1, P3, P4, P10). 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 aren't 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.

(C2) Mentors have to deal with newcomers' lack of holistic understanding about the project and its culture (mentioned by P1, P2, P3, P4, P9). 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."* These specifics may especially harm newcomers or casual contributors [48].

(C3) Lack of information about how newcomer-friendly a task is (mentioned by P1, P10). 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 small 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 (C1).

(C4) Difficulty in identifying the complexity of a task (mentioned by P7, P8). 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 Linux Kernel."* Therefore, developers most often rely on their experience to evaluate a task. However, experts suffer from the "curse of knowledge"—the difficulty in seeing something from an outsider's point of view [53]. 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."*

(C5) Difficulty in estimating the time necessary to finish a task. (mentioned by P1, P8). Complementary to the complexity of the tasks, mentors are also concerned about task duration. Usually, mentors do not want to give newcomers long-term assignments and, sometimes even low complexity tasks can take a long time. 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 and there aren't many committers to review. Also, sometimes, after a few months, the project has changed*

[16] https://doi.org/10.5281/zenodo.3970997

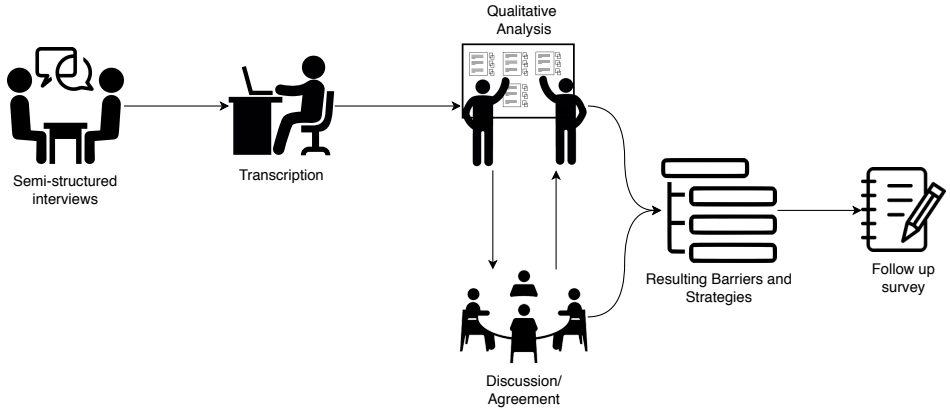

Figure 1: Research method overview

to the point where the originally submitted patch cannot even be used anymore."

(C6) Lack of newcomer-friendly tasks available (mentioned by P2, P4, P5). 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].*

(C7) Lack of available information about newcomer's skills, interests, and expertise (mentioned by P2, P4, P7, P9). 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."

4.2 What are the strategies applied by the mentors to recommend tasks? (RQ2)

We identified 13 strategies employed by mentors, as presented in Figure 2. We classified the strategies into five main categories, as described below. For each strategy, after their name we list the mentors who mentioned it.

4.2.1 Strategies to identify task complexity. As already mentioned, identifying task complexity is challenging, even for mentors (C4). 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. Although mentors rely on human judgment, they employ some strategies to support their analysis. We identified five strategies that mentors employ to identify task complexity (Table 3).

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

Table 3: Strategies to identify task complexity

Category	Strategy Name
Identify task complexity	Reproducing the bug
	Comparing the complexity of new bugs with existing bugs in bug tracker
	Tagging the task based on difficulty
	Adding documentation
	Discussing tagging in the conferences

Comparing the complexity of new bugs with existing bugs in the bug tracker (P2). 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 (P1, P2, P4, P5, P6, P7). Six mentors reported that tasks in their projects are tagged based on difficulty. Tags make newcomer-friendly tasks more visible to 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 3, 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 adds: *"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 shared 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. However, P10 reiterates 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."*

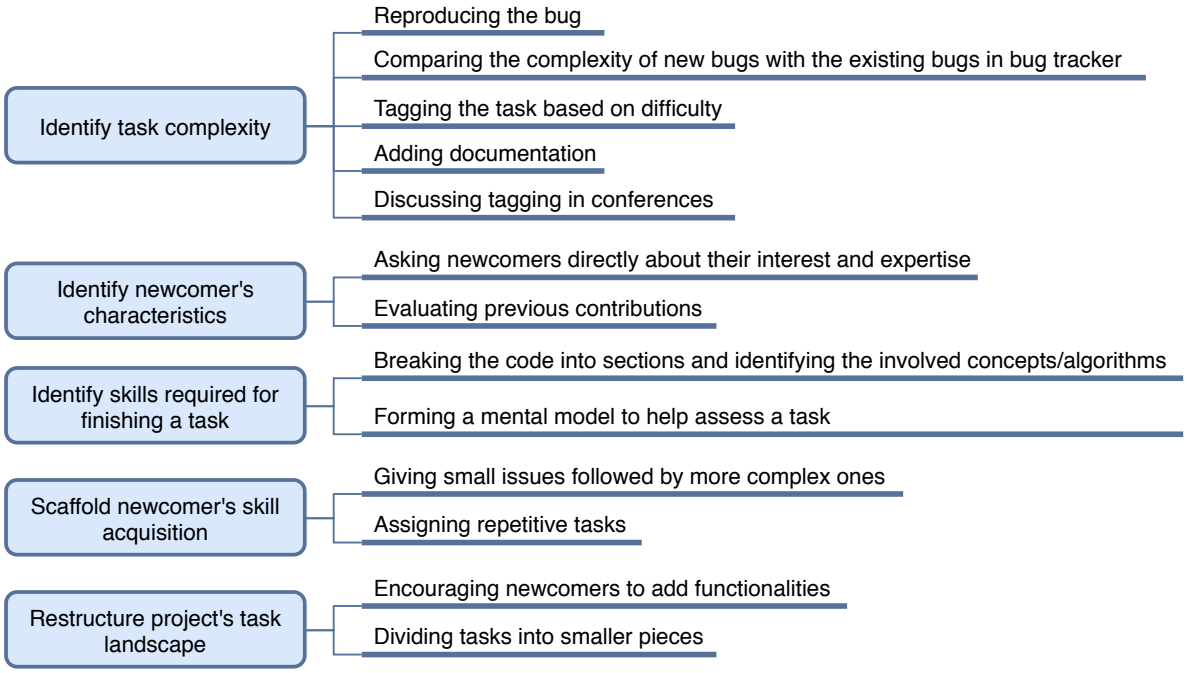

Figure 2: Overall view of the strategies

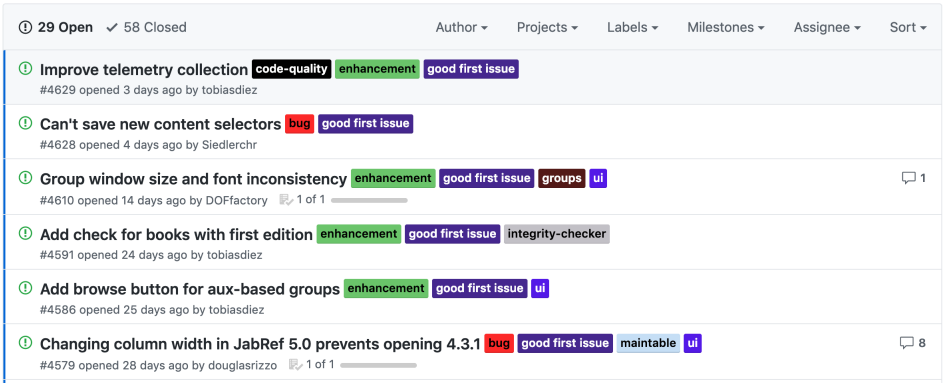

Figure 3: JabRef's issue tracker with labels ("good first issues") to help newcomers identifying appropriate tasks

However, as previously mentioned, the lack of information about how newcomer-friendly a task is (C3) was frequently cited as a challenge, and P10 points out 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 (P2, P4). 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 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 (P1). Due to the relevance of tagging tasks in OSS project environments, members of OSS projects hold discussions to improve tagging, as P1 explains: *"Some committers get together at conferences and meetups to discuss ideas to help newcomers when getting involved... We discuss tagging at those conferences."*

4.2.2 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 4.

Table 4: Strategies to identify the skills required to finish a task

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

Breaking the code into sections and identifying the involved concepts/algorithms (P5, P6, P10). 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 (P6, P10). We noticed that our participants reported forming a model of the project structure in their minds, which helped them in 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 [30, 31]. P10 reported that she thinks through a workflow model to consider the required skills and possible locations for a bug.

4.2.3 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 (C7). Eight out of ten of our participants mentioned they evaluate newcomers' expertise, and five out of ten mentioned they look for what newcomers are interested in before recommending tasks to them. Our interviewees highlighted that they avoid recommending tasks that newcomers will not like. As P3 states, *"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 5.

Table 5: Task recommendation strategies to identify the newcomers' characteristics

Category	Strategy Name
Identify newcomer's characteristics	Asking newcomers directly about their interests and expertise
	Evaluating previous contributions

Asking newcomers directly about their interests and expertise (P1, P3, P4, P6, P7, P9, P10). 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 complements 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 explains 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 [24]. 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 (P4, P9, P10). 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."*

4.2.4 Scaffolding newcomer's skill acquisition. Mentors frequently mentioned how they recommend a sequence of tasks to onboard newcomers, as presented in Table 6 and described below.

Table 6: Strategies to scaffold newcomers' skills acquisition

Category	Strategy Name
Scaffold newcomer's skill acquisition	Assigning a small task first and then challenging the newcomers with bigger tasks
	Recommending repetitive tasks

Assigning a small task first and then challenging the newcomers with bigger tasks (P3, P7). According to our interviewees, offering newcomers small starter tasks provides mentors with 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 barriers, 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 (P3). 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 complements this view: *"Usually, it's better if they don't have to learn a new skill right away."*

Letting newcomers choose their tasks (P1, P2, P4, P5, P6, P8, P9). 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 interests, as P5 explains: *"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."* Our interviewees (7 out of 10) mentioned that they usually identify a subset of tasks and allow newcomers to choose the tasks they are more willing to accomplish.

4.2.5 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 (C6). 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 7.

Table 7: Strategies to restructure the task's landscape

Category	Strategy Name
Restructure project's task landscape	Dividing tasks into smaller pieces
	Encouraging newcomers to add functionalities

Dividing tasks into smaller pieces (P2, P3). 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 (P5). 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."*

4.3 Which strategies help mitigate challenges when recommending tasks to newcomers? (RQ3)

Although some of the interviewees spontaneously reported some connections between strategies and challenges, this was not an explicit goal of our interviews. In the follow-up survey, as described in Section 3.1, we asked the respondents to identify which of the strategies they used to mitigate each of the challenges we identified; we also asked them (using open-ended questions) to provide any other strategies they used, and to mention the strategies that they have applied with success.

Table 8 presents the number of survey answers (out of 30) that link the strategies to the challenges. We present the cells as a heat map, in which the darker cells represent the most recurrent matches, while the lighter cells represent the least recurrent ones. Moreover, cells marked with * represent those for which the link also spontaneously appeared in the interviews. It is worth highlighting that each participant could check multiple strategies per challenge. From the table, it is possible to capture which kinds of strategies are applied to solve each challenge. For example, we can observe that to mitigate C4 (Difficulty in identifying the complexity of a task), the most applied strategies are tagging the task and comparing the new bugs with existing ones. We also observed that some challenges may be softened by multiple strategies (e.g., C2), while for others some specific strategies may be more appropriate than others (e.g., C6).

The table also helped us to explore a particular strategy's relevance. For example, the strategies "tagging the issues based on difficulty" and "adding documentation" are the most selected (77 and 85 times, respectively). Of the challenges these strategies help to overcome, one notices that the smallest number of matches for either strategy is 7. Tagging was the most mentioned strategy during our interviews (7 out of 10 mentors), which indicates that adding signs and more information to the issues may be a good way to help mentors and newcomers find appropriate tasks. Moreover, both were mentioned by 4 respondents (in the open questions) as successful ways to support task recommendation.

"Assigning a small task first and then challenge the newcomers with bigger tasks" and "dividing tasks into smaller pieces" were also selected several times (69 times each) as ways to overcome multiple challenges. Interestingly, when we asked participants through open questions which strategies they found to be the most successful, eight mentioned "start by assigning a small task, and then challenge the newcomers with bigger tasks." One of them explicitly noted that *"giving a small task to onboard the newcomer is the best."*

We also observed that some of the strategies were applied to specific challenges. The best examples are the strategies of "asking newcomers directly about their interests and expertise" and

Table 8: Strategies used to overcome challenges (C1–C7); the cells are shaded to reflect the number of times survey participants mentioned these mappings (darker cells mean more mentions). Each participant could check multiple strategies per challenge. Cells marked with * represent the matches identified in the interviews.

Strategy	C1	C2	C3	C4	C5	C6	C7
Reproducing the bug	1	15	15	4*	6	9	15
Comparing new bugs with the existing bugs in bug tracker	1	5	12	16*	10	5	3
Tagging the task based on difficulty	7	7*	11*	19*	12	13	8*
Adding documentation	11	19*	20*	12	7	9	7
Discussing tagging in conferences	11	7	10*	7	7	7	6
Breaking the code into sections and identifying the involved concepts/algorithms	7	15*	10	12	10	3	10
Asking newcomers directly about their interests and expertise	14	7	4	2	2	2	20*
Evaluating previous contributions	3	6	2	3	4	1	21*
Assigning a small task first and then challenge the newcomers with bigger tasks	15	19	3	8	7	7	10
Recommending repetitive tasks	11	14	3	2	4	7	12
Encouraging newcomers to add functionalities	8	14	3	2	0	5*	12
Dividing tasks into smaller pieces	4	11	9	13	12	15*	5

C1: Newcomers' social fear; **C2**: Lack of holistic understanding about the project; **C3**: Newcomers' lack of information about how newcomer-friendly a task is; **C4**: Difficulty in identifying the complexity of a task; **C5**: Difficulty in estimating the amount of time necessary to finish a task; **C6**: Lack of friendly tasks available for newcomers; **C7**: Lack of available information about newcomer's skills, interest, and expertise

"evaluating previous contributions," which heavily focus on finding information about newcomers' skills (to overcome C7).

In addition, we observed that the challenge of estimating the time necessary to finish a task (C5) is still open, with no strategy standing out as a way to help to overcome it. The most recurrent strategy for these challenges was "tagging the task based on difficulty." Finally, multiple strategies were selected by different respondents as a way to overcome the problems with newcomers' lack of a holistic understanding of the project (C2).

In general, the challenges with the highest number of matches in the survey are those that were also identified during the interviews. An exception is the category "assigning a small task first and then challenge the newcomers with bigger tasks." The respondents pointed out that it may help to overcome C1 (newcomers' experiencing social fears about working on challenging tasks) and C2 (newcomers' lack of holistic understanding about the project and its culture), but no explicit link emerged in the interviews. For the strategy "Reproducing the Bug," while half of the respondents mapped it to C2, C3, and C7, only four said that it helps to overcome C4 (difficulty in identifying the complexity of a task), which was the challenge to which the strategy was linked during the interviews. We hypothesize that this difference occurred because the survey was not explicit about who was responsible for reproducing the bug.

Lastly, we also asked the respondents to mention any other strategies that they use to recommend tasks to newcomers. Although we received 17 answers for this open question, the reported strategies could be classified into two types: strategies that are more general, related to onboarding or mentoring in general (not specifically for task recommendation); and replication or specialization of the strategies mentioned by our interviewees.

For the first type (strategies related to onboarding or mentoring in general), for example, one respondent mentioned *"daily quick meetings with newcomers"* as a good way to maintain consistent contact and follow-up with newcomers during the onboarding process (they could perhaps also recommend tasks during these meeting). In another case, a respondent offered a set of tips to support newcomers' attraction, training, and onboarding: *"[Help them to] get the software up and running with less effort; run coding workshops so people can work on issues together and support each other; give talks about our OSS, so people get curious."*.

For the second type (replication or specialization of the strategies mentioned by our interviewees), for example, people reported the usefulness of documentation (e.g., *"documentation is a big one"*); and the potential use of different kinds of tagging (*"continual tagging of difficulty levels for issues"*; *"tagging issues based on the skills and/or technologies that will be required to solve them."*; *"we tag and have identified mentors who are ready to assist"*. In other cases, people mentioned ways to assign small tasks and gradually increase complexity (e.g., *"ask newcomers to do those small tasks and when it comes to complex ones, we give them the chance to choose from a set of tasks."*; *"assign easy tasks to them. Real beginners might need to touch one class, plus test, only."*. Participants also mentioned ways to ask newcomers directly (e.g., *"personal conversation with the newcomer... that paints the newcomer and mentor in the same boat."* They confirmed that, through personal contact, *"usually newcomers show their interest in contributing to specific areas"*. In summary, the analysis of the answers to the open question did not reveal new categories of strategies; however, it provided insights that the strategies uncovered during the interviews are relatively comprehensive and cover different specific methods mentors use to recommend tasks.

5 DISCUSSION

In the section, we discuss task recommendation to newcomers in light of our results and related literature by proposing guidelines based on the identified challenges and strategies.

An overview of task recommendation by mentors. Finding an appropriate task to work in OSS projects is a hard task for newcomers [61]. This was also reported by Ann Barcomb and colleagues [7] in the context of episodic contributors. They found that community managers find it difficult to identify and maintain a list of tasks that they can be picked up by these contributors, who are essentially newcomers to the 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, usually not by relying 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 conferences. Besides identifying task complexity, mentors determine the skills required for finishing the tasks. Although some mentors are aware of tools that can help identifying the complexity of the tasks and the 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 and dividing tasks into smaller pieces.

Regarding the newcomers, 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 Ann Barcomb et al. [7]. For example, similar "Tagging the task based on difficulty," they report that "Identify appropriate tasks" and "Define one-off tasks" are good practices for community preparation to episodic contributors. Still, they identified "Detail how to complete a task" as a practice used to support the onboarding. Here we found indication that "Adding Documentation" helps to overcome most of the challenges faced when mentors recommend tasks to newcomers. This evidences that, while the strategies are not formally documented, they are widely used.

Provide easily accessible information about the 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 may be newcomer-friendly, or "low-hanging fruits" [74]. 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`[17] 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 4.2.1). Our survey respondents acknowledged that this strategy aids in overcoming most of the challenges (see Table 8). 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 (C3). 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 [60, 64]. Providing automated ways of tagging and better supporting human annotators are still open research opportunities that could provide great help to mentors and newcomers.

Explore different ways to understand newcomers' skills. Our results indicate that expertise and skill identification (C7) is a key challenge for mentors while recommending a task to newcomers. The evidence collected here extends the understanding that awareness of developers' skills is the foundation for building productive teams [23]. This result may also trigger several discussions and can be analyzed through the lens of the existing literature and theories. A recent study from 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.' [18], which was considered by their study as the most influential in how pleasant and efcient the integration experience was for the newcomer, and was also mentioned by 7 survey respondents as an effective way to reduce social fear (C1). 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." [24].

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) [49] and negotiation skills [34]. 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 [67] and likewise decrease communication effectiveness [12]. Thus, in some cases it may not reduce or eradicate newcomers' social fears [70], especially in the initial (or introductory) phase of the joining process [43]. Miscommunication on text-based channels is common, since context and expression is often lost [67]. 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 [66]. E-mentoring has a particular applicability for

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

OSS communities, as the work is conducted in a distributed way [71].

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. This is understandable since software development leaves traces of development activities in the repository, which can then be used for inferring the expertise of developers [17]. Existing tools such as "My GitHub Resume"[18] and "Visual Resume" [54] 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 [55]. One option 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 [29]. Instead of processing information in an isolated style, situated learning occurs though social interactions and puts a strong focus on practicing the acquired knowledge [19]. Fang et al. [22] 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.

Help newcomers develop a holistic view of the project. One challenge reported by mentors was that newcomers tend to lack a holistic understanding of the project (C2). This is a problem that has been previously mentioned in the literature. For example, Dagenais et al. [18] mention that newcomers are like explorers who must orient themselves within an unfamiliar landscape. Although this is not directly related to task recommendation, this challenge may have implications related to 'hidden' skills that the task may require (e.g., libraries, frameworks, social skills, and tools). Therefore, it is important that newcomers have a high-level view of the project so they can identify key components, tools, structures, processes, and practices and how they are related to the project. Providing upfront information is important and may guide newcomers by acting as maps and signs [63]. This was acknowledged by 19 survey respondents (63%), who mentioned adding documentation as a good way to overcome this challenge.

However, creating and maintaining this high-level (yet thorough) information may be challenging. In particular, summarizing and structuring information in a manner that does not overload the reader but still hits all relevant points is no easy feat. In addition, creating documentation aimed at newcomers may be challenging when diversity needs to be considered. Different newcomers may have different learning styles [13], requiring information to be displayed in various forms and depths. It is important to leverage information architecture techniques to design an appropriate way to display information to newcomers. Another challenge is maintaining documentation. According to the Open Source Survey [75], "incomplete or outdated documentation is a pervasive problem, observed by 93% of respondents, yet 60% of contributors say they rarely or never contribute to the documentation." Thus, although important, documentation is usually overlooked by the community. Failing to maintain documentation in the project may lead to untruthful or misleading information.

5.1 Future Opportunities and Implications

In this subsection, we discuss how different stakeholders can leverage our results. We also discuss gaps that were uncovered by our results and can be seen as potential future research streams.

OSS communities and 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. Moreover, we identified a set of task recommendation strategies and provided guidelines that can be used by mentors to recommend tasks to 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 [13] 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, as a continuation of this work, future work may focus on investigating how each strategy influences newcomers' motivation to work, retention in the project, and assertiveness in recommended tasks.

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

Education and Training Personnel. People interested in education and training can make use of our findings to identify effective ways to assign tasks to newcomers in OSS projects. Mentors play an important role in assigning tasks to newcomers and guiding them in finding appropriate tasks for themselves. When asked whether they had been trained to act as a mentor, all of our participants answered "no." Given the number of social barriers revealed by the participants, it is important that (future) professionals acquire the proper social skills that will better prepare them to mentor. Therefore, OSS communities and educators can devise specific training on the skills needed to be a mentor and on the best strategies to recommend tasks. For newcomers' education and training, the challenges evidenced here serve as a starting point for making instructors aware of what to expect when incorporating OSS projects in their teaching, which is becoming more common [11, 40, 47].

6 LIMITATIONS

Although we collected data from mentors with different backgrounds and continued interviewing until we were not able to identify any new challenges or strategies in 2 interviews in a row, we likely did not discover all possible challenges and strategies or provide complete explanation of them. To mitigate this, in our follow up survey we asked the mentors if they applied any additional strategies and no new strategy was found. Still, we are aware that the OSS universe is vast, and strategies can differ according to projects.

We acknowledge that the size of our samples can be considered small. However, 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 [10], and a minimum of six interviews for phenomenological studies [39]. Moreover, we reinforce that identifying people who are mentors in OSS projects is not trivial. Finding participants for this study was challenging because mentoring in such non-structured environments can take place through private, not publicly visible communication channels [20]. To increase the number of respondents in our study, we deployed multiple tactics to reach mentors, including 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 preferred not to reach people through project-specific mailing lists or communication channels to avoid spamming lists created for purposes not described in the code of conduct of the projects (which may lead to ethical concerns) [5]. Moreover, 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.

Since we employed a snowballing approach to sampling our participants, we acknowledge that sampling bias affects our interviewees' selection, 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, and adding the survey. Still, since our survey was anonymous, we cannot discard the potential intersection between survey respondents and interviewees. Although this is unlikely to happen, in the worst case our population was increased 200%. Moreover, even if interviewees responded to the survey, they were exposed to consolidated strategies, giving them a chance to reevaluate what was found.

Another potential threat to the results' validity is the subjectivity of the interview data classification. To avoid this threat, we used an approach in which all analysis was thoroughly grounded in the data collected and exhaustively discussed amongst the whole team to reach an agreement. The team includes researchers with extensive experience in qualitative methods, and precautions were taken to mitigate bias in the data collection, analysis, and report.

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 task assignment strategies in different domains to build generalized models and theories about onboarding and mentorship in open collaboration communities.

7 CONCLUSION

In this paper, we analyzed data collected through interviews and a survey conducted with mentors from different OSS communities to identify challenges faced and strategies used by mentors to recommend tasks to newcomers. In our analysis, we identified 7 challenges that mentors face when recommending tasks to newcomers. The identified challenges range from handling newcomers' low self-efficacy (C1) and understanding their background—with regard to the project (C2) and technical skills (C7)—to dealing with poor (or lack of) information about the tasks available at hand (C3-C6). Regarding the strategies, we identified 13 approaches to task recommendation. We further classified them into 5 umbrella categories: identify newcomers' characteristics, scaffold newcomers' skill acquisition, identify task complexity, identify skills required to finish a task, and restructure task landscape.

In our survey, we mapped the task recommendation strategies to the challenges faced by the mentors. Our analysis revealed that these strategies serve different goals (helping mitigating specific challenges), and some help dealing with multiple challenges, for instance, "Tagging the issues based on difficulty" and "adding documentation" are perceived as good approaches to dealing with multiple challenges that hinder task recommendation.

Scientifically identifying mentors' practices when recommending tasks for newcomers is, to the best of our knowledge, still unexplored research. Our results provide insights into how mentors perform these strategies and can be used by: newcomers and mentors to collaboratively devise strategies for skill acquisition and successful onboarding to OSS projects; mentors to guide their choices of strategies; OSS communities to improve the project's task landscape; and researchers and tool developers to design automated support for the strategies employed by OSS mentors.

ACKNOWLEDGEMENTS

We thank all interviewees and survey participants for their great contribution to this research. This work is partially supported by the National Science Foundation under Grant Numbers 1815486, 1815503, 1900903, and 1901031.

REFERENCES

[1] Tammy D Allen, Lillian T Eby, Georgia T Chao, and Talya N Bauer. 2017. 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 (March 2017), 324–337. https://doi.org/10.1037/apl0000086

[2] John Anvik and Gail C. Murphy. 2011. Reducing the Effort of Bug Report Triage: Recommenders for Development-oriented Decisions. *ACM Trans. Softw. Eng. Methodol.* 20, 3, Article 10 (Aug. 2011), 35 pages. https://doi.org/10.1145/2000791.2000794

[3] B. Ashok, Joseph Joy, Hongkang Liang, Sriram K. Rajamani, Gopal Srinivasa, and Vipindeep Vangala. 2009. 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)*. ACM, New York, NY, USA, 373–382. https://doi.org/10.1145/1595696.1595766

[4] Sogol Balali, Igor Steinmacher, Umayal Annamalai, Anita Sarma, and Marco Aurelio Gerosa. 2018. Newcomers' Barriers. . . Is That All? An Analysis of Mentors' and Newcomers' Barriers in OSS Projects. *Computer Supported Cooperative Work (CSCW)* 27, 3 (01 Dec 2018), 679–714. https://doi.org/10.1007/s10606-018-9310-8

[5] Sebastian Baltes and Stephan Diehl. 2016. Worse than spam: Issues in sampling software developers. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–6.

[6] Sebastian Baltes and Stephan Diehl. 2018. Towards a theory of software development expertise. In *ACM Symposium on the Foundations of oftware Engineering (FSE 2018)*. 187–200.

[7] Ann Barcomb, Klaas-Jan Stol, Brian Fitzgerald, and Dirk Riehle. 2020. Managing Episodic Volunteers in Free/Libre/Open Source Software Communities. *IEEE Transactions on Software Engineering* (2020).

[8] Andrew Begel and Beth Simon. 2008. Novice Software Developers, All over Again. In *Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/1404520.1404522

[9] Lucy M. Berlin. 1992. *Beyond Program Understanding: A Look at Programming Expertise in Industry*. Technical Report HPL-92-142. Hewlett-Packard Laboratories, Palo Alto, CA, USA. http://www.hpl.hp.com/techreports/92/HPL-92-142.html. Accessed in 18 February 2018.

[10] H Russell Bernard. 2017. *Research methods in anthropology: Qualitative and quantitative approaches*. Rowman & Littlefield.

[11] Judith Bishop, Carlos Jensen, Walt Scacchi, and Arfon Smith. 2016. How to Use Open Source Software in Education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 321–322. https://doi.org/10.1145/2839509.2844665

[12] Kevin Buffardi. 2017. 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*. 22–27.

[13] Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. 2016. 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)*. ACM, New York, NY, USA, 2586–2598. https://doi.org/10.1145/2858036.2858274

[14] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 2012. 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)*. ACM, New York, NY, USA, Article 44, 11 pages. https://doi.org/10.1145/2393596.2393647

[15] Gloria Crisp and Irene Cruz. 2009. Mentoring college students: A critical review of the literature between 1990 and 2007. *Research in higher education* 50, 6 (2009), 525–545. https://doi.org/10.1007/s11162-009-9130-2

[16] Davor Cubranic, Gail C. Murphy, Janice Singer, and Kellogg S. Booth. 2005. Hipikat: a project memory for software development. *IEEE Transactions on Software Engineering* 31, 6 (June 2005), 446–465.

[17] Jose Ricardo da Silva, Esteban Clua, Leonardo Murta, and Anita Sarma. 2015. 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)*. IEEE, 409–418.

[18] Barthélémy Dagenais, Harold Ossher, Rachel K. E. Bellamy, Martin P. Robillard, and Jacqueline P. de Vries. 2010. Moving into a New Software Project Landscape. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10)*. ACM, New York, NY, USA, 275–284. https://doi.org/10.1145/1806799.1806842

[19] Amy Edmondson. 1999. Psychological Safety and Learning Behavior in Work Teams Amy Edmondson. *Administrative Science Quarterly* 44, 2 (1999), 350–383.

[20] Fabian Fagerholm, Alejandro S. Guinea, Jürgen Münch, and Jay Borenstein. 2014. 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)*. ACM, 55:1–55:10. https://doi.org/10.1145/2652524.2652540

[21] Fabian Fagerholm, Patrik Johnson, Alejandro Sanchez Guinea, Jay Borenstein, and Jürgen Münch. 2014. Onboarding in Open Source Projects. *IEEE Software* 31, 6 (Nov. 2014), 54–61. https://doi.org/10.1109/MS.2014.107

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

[23] Samer Faraj and Lee Sproull. 2000. Coordinating expertise in software development teams. *Management science* 46, 12 (2000), 1554–1568.

[24] N. T. Feather. 1969. 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). https://doi.org/10.1037/h0028071

[25] Jane Forman and Laura Damschroder. 2007. Qualitative content analysis. *Empirical methods for bioethics: A primer* 11 (2007), 39–62.

[26] Andrea Forte and Cliff Lampe. 2013. Defining, Understanding, and Supporting Open Collaboration: Lessons From the Literature. *American Behavioral Scientist* 57, 5 (2013), 535–547. https://doi.org/10.1177/0002764212469362 arXiv:https://doi.org/10.1177/0002764212469362

[27] D Randy Garrison, Martha Cleveland-Innes, Marguerite Koole, and James Kappelman. 2006. Revisiting methodological issues in transcript analysis: Negotiated coding and reliability. *The Internet and Higher Education* 9, 1 (2006), 1–8.

[28] Susan Gershenfeld. 2014. A Review of Undergraduate Mentoring Programs. *Review of Educational Research* 84, 3 (2014), 365–391. https://doi.org/10.3102/0034654313520512 arXiv:https://doi.org/10.3102/0034654313520512

[29] Karen Handley, Andrew Sturdy, Robin Fincham, and Timothy Clark. 2006. Within and beyond communities of practice: Making sense of learning through participation, identity and practice. *Journal of management studies* 43, 3 (2006), 641–653.

[30] P. N. Johnson-Laird. 1983. *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Harvard University Press, Cambridge, MA, USA.

[31] Philip N. Johnson-Laird. 2010. Mental models and human reasoning. *Proceedings of the National Academy of Sciences* 107, 43 (2010), 18243–18250. https://doi.org/10.1073/pnas.1012933107 arXiv:http://www.pnas.org/content/107/43/18243.full.pdf

[32] Swapna Kumar, Melissa Johnson, and Truly Hardemon. 2013. Dissertations at a distance: Students perceptions of online mentoring in a doctoral program. *International Journal of E-Learning & Distance Education* 27, 1 (2013). http://www.ijede.ca/index.php/jde/article/view/835

[33] Adriaan Labuschagne and Reid Holmes. 2015. Do Onboarding Programs Work?. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*. IEEE Press, Piscataway, NJ, USA, 381–385. https://doi.org/10.1109/MSR.2015.45

[34] Timothy C Lethbridge. 2000. What knowledge is important to a software professional? *Computer* 33, 5 (2000), 44–50.

[35] Craig Macdonald and Iadh Ounis. 2006. 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*. ACM, 387–396.

[36] Yuri Malheiros, Alan Moraes, Cleyton Trindade, and Silvio Meira. 2012. A Source Code Recommender System to Support Newcomers. In *Proceedings of the IEEE 36th Annual Computer Software and Applications Conference (COMPSAC '12)*. IEEE, Los Alamitos, California, USA, 19–24. https://doi.org/10.1109/COMPSAC.2012.11

[37] Kay Martinez. 2004. Mentoring New Teachers: Promise and Problems in Times of Teacher Shortage. *Australian Journal of Education* 48, 1 (2004), 95–108. https://doi.org/10.1177/000494410404800107 arXiv:https://doi.org/10.1177/000494410404800107

[38] Ann Mihkelson. 1997. A Model of Research Mentoring for Higher Education–An Overview. (1997).

[39] Janice M Morse. 1994. Designing funded qualitative research.. In *Handbook of qualitative research*. Sage Publications, Inc, 220–335.

[40] Debora Nascimento, Kenia Cox, Thiago Almeida, Wendell Sampaio, Roberto Bittencourt, Rodrigo Souza, and Christina Chavez. 2013. Using Open Source Projects in software engineering education: A systematic mapping study. In *IEEE Frontiers in Education Conference*. IEEE, 1837–1843. https://doi.org/10.1109/FIE.2013.6685155

[41] Katherine E Nugent, Gwen Childs, Rosalind Jones, and Pamela Cook. 2004. A mentorship model for the retention of minority students. *Nursing Outlook* 52, 2 (2004), 89–94. https://doi.org/10.1016/j.outlook.2003.09.008

[42] Gordon O'Brien. 1967. Methods of Analyzing Group Tasks. (1967), 1–40. https://doi.org/dtic/tr/fulltext/u2/647762.pdf

[43] Ilan Oshri, Julia Kotlarsky, and Leslie P Willcocks. 2007. Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects. *The Journal of Strategic Information Systems* 16, 1 (2007), 25–49.

[44] Sebastiano Panichella. 2015. Supporting newcomers in software development projects. In *IEEE International Conference on Software Maintenance and Evolution (ICSME 2015)*. IEEE, 586–589. https://doi.org/10.1109/ICSM.2015.7332519

[45] Yunrim Park and Carlos Jensen. 2009. Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In *Proceedings of the 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT '09)*. IEEE, 3–10.

[46] Stephanie C Payne and Ann H Huffman. 2005. A longitudinal examination of the influence of mentoring on organizational commitment and turnover. *Academy of Management Journal* 48, 1 (February 2005), 158–168. https://doi.org/10.5465/

AMJ.2005.15993166

[47] Gustavo Pinto, Igor Steinmacher, Fernando Figueira Filho, and Marco A. Gerosa. 2017. 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)*. IEEE, Los Alamitos, California, USA, 5.

[48] Gustavo Pinto, Igor Steinmacher, and Marco Gerosa. 2016. More Common Than You Think: An In-Depth Study of Casual Contributors. In *SANER*. 112–123.

[49] Alex Radermacher and Gursimran Walia. 2013. Gaps between industry expectations and the abilities of graduates. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 525–530.

[50] Belle Rose Ragins. 1989. Barriers to Mentoring: The Female Manager's Dilemma. *Human Relations* 42, 1 (January 1989), 1–22. https://doi.org/10.1177/001872678904200101

[51] Donna Redman, Sharon Conley, and Terrence E. Deal. 2015. A cultural approach to mentoring new teachers. In *Mentoring for school quality: How educators can be more professional and effective*, Bruce S. Cooper and Carlos R. McCray (Eds.). Rowman & Littlefield, Lanham, Maryland, 65–80.

[52] Jonah E Rockoff. 2008. *Does mentoring reduce turnover and improve skills of new employees? Evidence from teachers in New York City*. Technical Report. National Bureau of Economic Research.

[53] Neil J Salkind. 2008. *Encyclopedia of educational psychology*. Sage Publications.

[54] Anita Sarma, Xiaofan Chen, Sandeep Kuttal, Laura Dabbish, and Zhendong Wang. 2016. Hiring in the Global Stage: Profiles of Online Contributions. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. 1–10. https://doi.org/10.1109/ICGSE.2016.35

[55] Anita Sarma, Marco Aurélio Gerosa, Igor Steinmacher, and Rafael Leano. 2016. Training the Future Workforce Through Task Curation in an OSS Ecosystem. In *24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. ACM, New York, NY, USA, 932–935. https://doi.org/10.1145/2950290.2983984

[56] Andreas Schilling, Sven Laumer, and Tim Weitzel. 2012. 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)*. IEEE Computer Society, 3446–3455. https://doi.org/10.1109/HICSS.2012.644

[57] Jefferson Silva, Igor Wiese, Daniel German, Christoph Treude, Marco Gerosa, and Igor Steinmacher. 2020. 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)*.

[58] Susan E. Sim and Richard C. Holt. 1998. 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)*. IEEE, 361–370. https://doi.org/10.1109/ICSE.1998.671389

[59] Donna Spencer. 2009. *Card sorting: Designing usable categories*. Rosenfeld Media.

[60] Igor Steinmacher, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Awareness Support in Distributed Software Development: A Systematic Review and Mapping of the Literature. *Computer Supported Cooperative Work (CSCW)* 22, 2-3 (2013), 113–158. https://doi.org/10.1007/s10606-012-9164-4

[61] Igor Steinmacher, Tayana Conte, and Marco Aurélio Gerosa. 2015. 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)*. IEEE, 5299–5308.

[62] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David F. Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 1379–1392. http://doi.acm.org/10.1145/2675133.2675215

[63] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming Open Source Project Entry Barriers with a Portal for Newcomers. In *38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 273–284. https://doi.org/0.1145/2884781.2884806

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

[65] Igor Steinmacher, Igor Scaliante Wiese, and Marco Aurélio Gerosa. 2012. Recommending mentors to software project newcomers. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering (RSSE '12)*. IEEE Computer Society, Washington, DC, USA, 63–67. https://doi.org/10.1109/RSSE.2012.6233413

[66] Heidrun Stoeger, Xiaoju Duan, Sigrun Schirner, Teresa Greindl, and Albert Ziegler. 2013. The effectiveness of a one-year online mentoring program for girls in STEM. *Computers & Education* 69 (2013), 408–418.

[67] Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M German. 2016. How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering* 43, 2 (2016), 185–204.

[68] Anselm Strauss and Juliet M. Corbin. 2007. *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory* (3rd ed.). SAGE Publications.

[69] Chris Street. 2004. Examining Learning To Teach through a Social Lens: How Mentors Guide Newcomers into a Professional Commuity of Learners. *Teacher Education Quarterly* 31, 2 (2004), 7–24.

[70] Gil Taran and Lynn Carter. 2010. Improving distance mentoring: Challenges and how to deal with them in global development project courses. In *Conference on Software Engineering Education and Training (CSEET)*. IEEE, 97–104.

[71] Erik H Trainer, Arun Kalyanasundaram, and James D Herbsleb. 2017. 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)*. IEEE, 107–116.

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

[73] Jianguo Wang and Anita Sarma. 2011. 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)*. ACM, New York, NY, USA, 76–79.

[74] Vincent Wolff-Marting, Christoph Hannebauer, and Volker Gruhn. 2013. 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)*. IEEE, 9–14. https://doi.org/10.1109/SoMeT.2013.6645669

[75] Frances Zlotnick. 2017. GitHub Open Source Survey 2017. http://opensourcesurvey.org/2017/. (jun 2017). https://doi.org/10.5281/zenodo.806811