2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) | 978-1-6654-3786-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SANER53432.2022.00032

Hits and Misses: Newcomers' ability to identify Skills needed for OSS tasks

Italo Santos*, Igor Wiese*†, Igor Steinmacher*†, Anita Sarma‡ and Marco A. Gerosa*
*Northern Arizona University, Flagstaff, AZ, USA
†Federal University of Technology, Campo Mourao, PR, Brazil
‡Oregon State University, Corvallis, OR, USA
Email: italo_santos@nau.edu, igor@utfpr.edu.br, igorfs@utfpr.edu.br, anita.sarma@oregonstate.edu, marco.gerosa@nau.edu

***Abstract*—Participation in Open Source Software (OSS) projects offers real software development experience for students and other newcomers seeking to develop their skills. However, onboarding to an OSS project brings various challenges, including finding a suitable task among various open issues. Selecting an appropriate starter task requires newcomers to identify the skills needed to solve a project issue and avoiding tasks too far from their skill set. However, little is known about how effective newcomers are in identifying the skills needed to resolve an issue. We asked 154 undergrad students to evaluate issues from OSS projects and infer the skills needed to contribute. Students reported a total of 94 skills, which we classified into 10 categories. We compared the students' answers to those collected from 6 professional developers. In general, students misidentified and missed several skills (f-measure=0.37). Students had results closer to professional developers for skills related to database, operating infrastructure, programming concepts, and programming language, and they had worse results in identifying skills related to debugging and program comprehension. Our results can help educators who seek to use OSS as part of their courses and OSS communities that want to label newcomer-friendly issues to facilitate onboarding of new contributors.**

***Index Terms*—Open-source software, Newcomers, Skills, Expertise.**

I. Introduction

To provide students experience with real software development problems, educators are increasingly using Open Source Software (OSS) projects as a training ground [1]–[4]. There are many OSS projects to choose from, ranging in different domains, sizes, and complexity [5], [6], giving educators a wide range of projects to train their students on. Such use of OSS as a training ground allows students to not only learn real-world technical skills but also learn about team communication, communication styles, and attitudes, which might, in turn, increase their confidence when applying for industry jobs [2], [7]–[9]. Successful participation in OSS projects also helps (student) newcomers gain visibility among their peers [10], [11], benefit society by developing a product used by many users [12], and have a higher chance to achieve professional success [11], [13], [14].

However, newcomers face a plethora of challenges [15], including choosing a task to start contributing [16], [17]. Newcomers are expected to be able to find a task on their own that they can complete. They have to figure out which task is appropriate from a set of open issues that require different skills and are of varying complexity. However, it is not easy to infer the skills (or expertise) required for a task solely from the task description. And very few projects annotate tasks to signal their appropriateness for newcomers. Skills in this context can represent technical knowledge or knowledge about the contribution process. When the gap between newcomers' skills and those needed to accomplish the task is too wide, it demotivates them causing them to dropout [16], [18], [19]. This particularly impacts students, who typically have a limited skill set and experience when first contributing to an OSS project.

So far, little is known about how effective (newcomer) students are in identifying the skills needed to resolve an issue based only on the information available on the issue tracker. In this paper, we seek to understand how effective students are in identifying the skills needed to work on an issue through the following research questions:

> ***RQ.1:*** How similar are the skills identified by students and professional developers when analyzing OSS issues?
>
> ***RQ.2:*** Which types of skills are students capable of identifying?

To answer these research questions, we tasked 154 undergraduate students to evaluate issues from 47 OSS projects and report which skills they considered necessary to close the issue. We recruited six professional software developers to assess the issues reported by the students. Then, we compared professional and students' responses using traditional information retrieval measures: recall, precision, and F-measure.

Our contributions in this paper include:

- characterization of how capable students are in identifying skills needed to work on OSS issues as compared to professional developers;
- identification and classification of a set of 94 skills into 10 higher-level categories;
- identification of the categories of skills where students are better (e.g., database, operating infrastructure, programming concepts, and programming language) or worse (e.g., debugging and program comprehension);

These results are particularly relevant for educators who use OSS in their courses, OSS communities that want to

978-1-6654-3786-8/22/$31.00 ©2022 IEEE
DOI 10.1109/SANER53432.2022.00032

label the issues to facilitate new contributors' onboarding, and researchers who aim to propose technical approaches to help identify skills in issues.

II. Related Work

OSS in education: Bringing OSS projects into the context of a classroom has been studied from diverse perspectives [6], [8], [20]. Smith et al. [20] reported the search for suitable OSS projects to teach an introductory SE course with a focus on maintenance and evolution. Morgan and Jensen [8] detailed the experience of teaching a SE course based on OSS projects. Their work compares and contrasts two different models and discusses the outcomes, lessons learned, and guidance to those developing their courses on this topic. Pinto et al. investigate the benefits, challenges, and opportunities from the professor's [3] and student's [2] perspectives. With our work, we want to understand the difficulty newcomers have in identifying skills from OSS task descriptions in the issue tracker, which is a critical part of the contribution process.

Onboarding newcomers: A newcomer is a developer trying to place their first code contributions into the project [21]. Newcomers often face hostile and unfamiliar landscapes when onboarding to an OSS project. According to Fogel [22], if a project does not make a good first impression, newcomers may wait a long time before giving it a second chance. This is especially pertinent for most students in software engineering classes as they are still novices and are developing their skills. Newcomers need proper orientation to navigate the project and correctly make a contributions [23]. Motivating, engaging, and retaining new developers in a project is essential to sustain a healthy OSS community [24]. Dagenais et al. [25] compare newcomers to explorers in a hostile environment where they need to self-guide through the tasks and obstacles in OSS.

Several empirical studies have focused on how newcomers join community-based OSS projects [17], [19], [21], [26], [27]. Other works have focused on understanding the barriers that influence newcomers' onboarding experiences [16], [28]. Developers indicated that the lack of awareness and guidance during their first steps (setup and choosing the right starter task) discouraged further contributions in OSS projects [29]. Researchers [16], [18], [30] have reported problems associated with the "difficulty to find a task to start with." Being able to identifying the right skills required to work on an issue can help newcomers find the right task and be successful in their first contribution. Our work investigates this topic deeper, identifying to what extent students can identify skills as compared to professional developers.

Skill identification: In our context, a skill is the knowledge needed for a newcomer to solve a task in an OSS project. Identifying skills is important ti choose a task [16], [19]. Some studies propose solutions to support task selection, including automated recommendation systems [31], [32]. Anvik and Murphy [31] use machine learning in the project history to recommend the expert for a given artifact. Macdonald and Ounis [32] apply data fusion techniques using a voting heuristic-based approach to analyze the change history of artifacts related to a task. Balasubramanyan et al. [33] propose a search tool called DebugAdvisor, which allows users to search through software repositories to recommend developers based on expertise on the source code related to the task. Costa et al. [34] identify the expert most suited to merge changes based on past work. However, these systems only suggest tasks to developers who have previous interactions in the project, and thus cannot support newcomers. Newcomers would need additional help to choose an appropriate set of tasks. Research thus far has not focused on models that articulate the skills necessary to contribute to an OSS project or on how to model skill acquisition trajectories.

Other studies focused on tools to support newcomers' onboarding. Čubranić et al. [35], for example, developed a tool called Hipikat used to assist newcomers by building a group memory and recommending source code, mail messages, and bug reports. Park and Jensen [17] showed that visualization tools could support the first steps of newcomers to OSS projects, helping them to find information more quickly. Wang and Sarma [19] created an approach to enable newcomers explore the socio-technical dependencies and the resources needed to fix a bug by providing information about similar past bugs. Steinmacher et al. [23] proposed and evaluated FLOSScoach, a web portal created to support the first contributions of newcomers to OSS projects. Results indicate that FLOSScoach played an important role in guiding newcomers and in lowering barriers related to the orientation and contribution process, whereas it was not effective in lowering technical barriers.

To complement the existing literature, in our work, we investigate how accurate are newcomer students in identifying skills from open source issues (as compared to professional developers) and indicate the skills categories that newcomers have more difficulty in identifying.

III. Research Method

We answer the research questions in this paper (see Section I) through the following set of five activities, as illustrated in Figure 1. In this study, we focus on undergraduate students, as educators have been using OSS to train students and these students are potential OSS project contributors [23]. In fact, multiple programs (e.g., Google Summer of Code [36], Facebook Open Academy) focus on attracting students to open source.

Activity 1 - Identification of skills by students. We recruited 154 undergraduate students who were junior or senior and had sufficient knowledge to fix bugs in software projects. They were enrolled in four different editions of software engineering courses from two different universities between 2016 and 2019. We asked the students to choose any OSS project from OpenHub[1] and *select one issue*. In total, the 154 students chose a variety of 47 OSS projects. After they selected the project and an issue, we asked them an open question: "Based only on the description of the issue and your knowledge about the

[1]https://www.openhub.net/

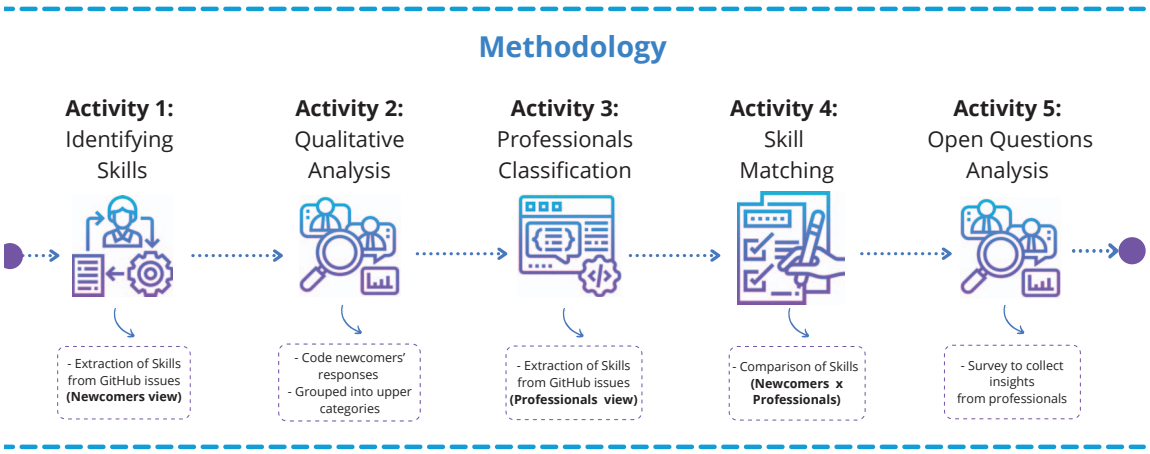

Figure 1. Research method overview.

project, what skills do you believe are necessary to solve the issue?"

Activity 2 - Qualitative analysis of skills: Based on the answers from Activity 1, we performed a qualitative analysis of the skills reported by the students. Before our analysis, we cleaned the data to remove invalid responses. We removed seven responses from the dataset because they either: (i) did not provide clear information about the skills required to solve the issue (4 responses), e.g. "*This bug is very complex and a large team is required to fix it.*"; or (ii) they were empty (3 responses). In the end, we received 147 valid responses that correspond to 131 different OSS issues. The issues and the responses are available in the supplementary material[2].

The analysis process started by selecting a random sample of issues in the dataset, and two of the authors independently coded the students' responses and grouped them into higher-level categories, following the open coding procedure [37]. The researchers evaluated a small set of responses, then compared and discussed the labels, to establish a common understanding. After three rounds of labeling and discussion the researchers independently categorized the 147 responses and got an agreement of 92.3%. The disagreements were individually discussed until finding consensus.

Then, we grouped the skills identified in the higher level categories. For example, we grouped skills such as, C, C++, Java, Python as "programming language". In total, we identified 94 different skills and categorized them into the following 10 categories: database, debugging, external libraries, operating infrastructure, program comprehension, programming concepts, programming language, project architecture, project-specific concepts, and testing.

Activity 3 - Identification of skills by professionals: In this step, we provided to professionals the same issues analyzed by the students and asked them to identify the skills required to solve the issues. We asked the professionals to evaluate only the projects/issues in which they had confidence or had worked in the past. Afterwards, we asked professionals to assess the students' responses and fill out a form to provide feedback about the analysis performed.

We recruited 6 professionals, from well-established and widely-used OSS projects, who had at least two years of experience. To recruit the professionals, we first searched for contributors who worked on the projects related to the issues chosen by the students. After a professional completed the study task, we asked them to recommend another qualified OSS contributor for us to contact. All of the professionals received a gift card as a token of appreciation for their participation.

The participants represent diverse OSS projects and have distinct backgrounds. As shown in Table I, the professionals who participated in our study had on average more than ten years of experience with software development and expertise with languages such as Java, JavaScript, Python, R, Lua, C, SQL, Delphi, Cobol, Pascal, among others. All the professionals had experience contributing to OSS projects (e.g., JabRef, Linux Kernel, Smatch, Dolphin emulator, Burger, Audacity). We also asked them their area of expertise. Responses show a diverse range of knowledge within the software development spectrum (e.g., programming, management, software testing, software architecture/design, software optimization, static analysis, debugging, reverse-engineering, and databases).

Activity 4 - Skill matching: To evaluate the skills identified by students, we compared them with the professionals' responses. To this end, we used traditional information retrieval measures: recall, precision, and F-measure [38]. To calculate these metrics we considered the data from the professionals as our ground truth.

Recall and precision enable us to evaluate the matches between the skills identified by students and our ground truth (i.e., professionals' responses). If the precision is low, the skills reported by students present many false positives. If the recall

[2]https://zenodo.org/record/5574248#.YXC_ytnMJb8

Table I
PROFESSIONALS DEMOGRAPHICS

ID	Gender	Age	Educational Level	Years of Experience		OSS Project		Professionals Expertise
				OSS	Industry	Role	Name	
P1	M	41-50	Bachelor	19	19	Developer	Linux Kernel, Smatch	Linux kernel, static analysis
P2	M	20-25	Bachelor	4	6	Developer	JavaScript	Software architecture/design, Software optimization, Testing
P3	M	Above 50	MSc degree	3	30	Contributor	JabRef, NAU-OSL	Programming, Databases
P4	M	20-25	Bachelor	6	8	Contributor	Dolphin emulator, Burger (reverse-engineering tool)	Programming, Debugging, Reverse-engineering
P5	M	31-40	Bachelor	2	12	Tech Lead	Audacity	Programming, Technical management
P6	M	26-30	Bachelor	2	6	Developer	JabRef, NAU-OSL	Programming, Leading, Testing

is low, there are too many false negatives. The F-measure is the harmonic mean of precision and recall. We calculate the F-measure using the following formula:

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Activity 5 - Open questions analysis: To collect insights about how professionals identified the skills, we asked them the following questions:

1) How did you choose tasks that fit your expertise?
2) What advice do you recommend for newcomers when they have to identify skills from GitHub issues? What information should they look at?
3) How did you identify the skills from GitHub issues?
4) Which issues from the categories defined in our research do you think are more difficult to identify?

The data gathered in the survey was quantitatively and qualitatively analyzed. Each survey response was analyzed by one author and reviewed by the other authors. We grouped the professionals' responses into main topics, applying open coding to classify and gain insights about how professionals identified skills from issue descriptions.

IV. RESULTS

In this section we present the results of our study, answering the RQs and presenting the analysis of the feedback from the professional developers.

A. *RQ.1. How similar are the skills identified by students and professional developers when analyzing OSS issues?*

Our data shows that students identify skills with a precision of 36% and recall of 38%. We can observe from these results that the students' provide a non-negligible number of false negatives and positives. These results show that students have a hard time identifying the skills in OSS issues.

Additionally, we calculated the *Hamming loss*[3] metric to measure the fraction of the wrong skills identified as compared to the total number of identified skills. Hamming loss metric has been originally used to evaluate text classification algorithms. It calculates the average number of times a label in the test set is incorrectly classified, including cases where an event (here skill) is missing its correct label (e.g., debugging) or if an instance has been incorrectly associated with a "wrong" label (e.g., database) [39]. The smaller the value of Hamming loss, the greater the similarity between the two label sets and the better the classification performance. The hamming loss metric results show that students incorrectly identified 21% of the skills.

[3]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming_loss.html

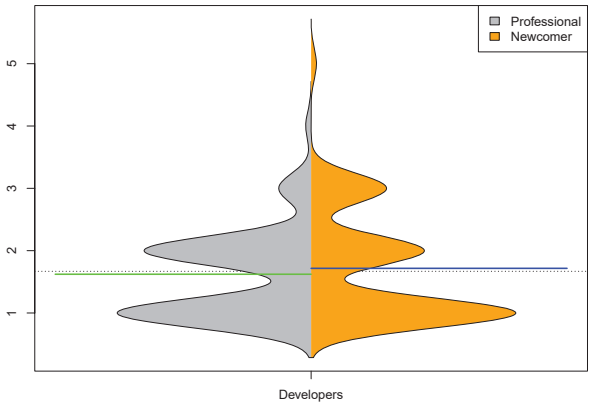

Figure 2. Distribution of skills categories.

We then calculated the exact match ratio, which is a strict measure of the classification performance—counting the number of issues that have exact label matches (with no false positives or false negatives). A drawback of this measure is that is does not consider issues with partially correct labels, which in our case would result if students identified only one skill opposed to two skills identified by professionals. In our case, the exact match rate was 10% (15 issues out of 148). We compare the distribution of the number of skills identified by students and professionals per issue. We present the distribution using the violin plot in Figure 2.

Our results also show that in 73 issues (49% of our data set), students had not been able to identify any correct skills. These results highlight the severe challenge that students face when trying to identify the skills needed to solve an issue. We also have a higher number of false positives (162 occurrences on 22 issues) that occurred when students report incorrect skills. This means that students may misjudge their ability to solve an issue if they incorrectly identify the skill needed, and become frustrated or disillusioned when trying to solve the issue when they do not possess the skills needed for the task. Students

also had false negatives (148 occurrences on 17 issues), that is, cases where students did not identify skills reported by the professional.

> ***RQ.1 Summary.*** Students' overall performance compared to professionals' is subpar. They achieved low results—precision (36%), recall (38%), and hamming loss (21%)—in correctly identifying the skills needed to complete an issue in OSS projects.

B. *RQ.2 - Which types of skills are students capable of identifying?*

Students identified 94 different types of skills further classified into 10 categories (see Table II).

The skill categories that were most frequently reported by students were *operating infrastructure* (25 instances) and *programming concepts* (22 instances). Other skills mentioned by students comprise the knowledge required about *programming languages* (10 instances) to start contributing to an OSS project and *program comprehension* (5 instances) to help in the understanding of the code, how it behaves, and how the student can assist in the project evolution. The need to understand the project was also pointed out by students when they identified skills related to *project architecture* (8 instances) and *project-specific concepts* (5 instances). Other skills categories related to *testing* (3 instances) and *debugging* (2 instances) were reported as relevant skills in solving specific types of issues. Finally, skills regarding how to use *databases* (3 instances) to manage OSS project data and how to leverage *external libraries* (9 instances) to improve software performance were also mentioned.

We also calculated the frequency with which issues were tagged with skill categories (Figure 3). As per the students, there were 56 times when issues were tagged with "programming language" skills and 55 times with "operating infrastructure" skills. While compared with professionals' view, the most frequent skill categories are *programming concepts* (77 times), *programming languages* (60 times), and *operating infrastructure* (55 times). Therefore, we posit that students who want to contribute to OSS projects should know the programming concepts and programming language in which the project was built, and learn some aspects that involve the environment in which the software was developed.

Table III presents the results from precision and recall grouped by skill category. For instance, professionals identified 26 issues in the database category, whereas students identified 8 issues, out of which only 6 were correct, as shown in the *correct* column. Five categories show zero responses (e.g., external libraries, program comprehension, project architecture, project specific concepts, and testing); this happened because even if the student identified an issue from one of those categories, their response was incorrect when compared with the same response from the professional. For example, in the project architecture category, professionals indicated that only 1 issue in our data set is related to this category compared with students' responses that show 27 issues from that category. None of the issues indicated by students in this category are correct when compared to professionals' classification.

In general, our results indicate that students do not perform well in identifying skills from issues in OSS projects. The results precision of the classification varied between 0.12 to 0.75 and recall between 0.50 to 0.58. The categories where students had better success were: *database, operating infrastructure, programming concepts, and programming language*. However, our manual evaluation of the issues in these categories revealed that these issues had explicit mentions to concepts related to databases (e.g., MySQL, tables, columns), the software environment necessary to develop the software (e.g., operating systems, git), or explicitly mentioned specific programming concepts (e.g., object orientation, data structures) or the specific programming language used to build the software (e.g., Java, PHP, Python).

Our results show that students have difficulty in identifying issues in the categories of *debugging* and *program comprehension*. These categories highlight software analysis concepts to detect and remove possible bugs and skills related to software behavior (e.g., code comprehension, static analysis).

We did not have sufficient data on issues related to the following five categories: external libraries, program comprehension, project architecture, project-specific concepts, and testing). However, our data shows that students often incorrectly identify skills from these categories. For example, in an issue from *MySQL*[4] project, the student identified the skill required as "testing", but the professional identified the skills of database and programming concepts (and did not mention testing).

> ***RQ.2 Summary.*** Our results suggest that students do not perform well in identifying skills from issues in OSS projects. Their responses frequently diverge from professionals. Students were better able to identify skills about: *database, operating infrastructure, programming concepts, and programming language*, and worse about *debugging* and *program comprehension*.

C. *Professionals feedback*

After we asked the professionals to identify skills in the same issues that the students analyzed, we also asked them to complete a survey to get their feedback about their experience in selecting tasks to work on. We discuss their responses next.

Task Selection. Professionals answered the survey question: *How do you choose tasks that fit your expertise?* Table IV presents their responses. Professionals reported that they select issues that are related to the programming language they are conformable with. As mentioned by P2, "*I selected projects that utilize the languages I know best.*"

However, professionals also tend to select issues that represent an exciting challenge to improve and hone their skills.

[4]https://bugs.mysql.com/bug.php?id=24762

Table II
SKILLS MODELLING CATEGORIES

Skills Total	Categories	Definition	Examples
3	Database	A collection of data stored and accessed electronically from a computer system.	MySQL, RQG tests, SQL
2	Debugging	Process of finding and resolving defects or problems within a computer program.	Code tracing, Reproduce the bug
9	External Libraries	Related to the libraries used by the software or in the development process.	DX10, DX11, ggit-1.0 library, glib-2.0 library, Gravatar, GTK.ScrolledWindow, GTK+, GTKBuilder, Libmetis libraries, Redshift GTK
25	Operating infrastructure	Related to aspects that involve hardware and software environment in which the software will run as well as installation and deployment.	Apache server, Compilers, Deployment, Git, GNOME, Hardware components, HTTP protocols, Input/Output, Installation, Internet of things, Linux, Linux Kernals security's common capability, Mac OS X, Memory (comsuption, handling, mapping input/output), Network, NPROC, Operating systems design, Preloaded Public Key Pinning, Runtime monitoring, Security, Software backup files, Ubuntu, wxGTK 3.0x
5	Program comprehension	It is the activity of understanding software code and its behavior.	Code comprehension, Code parsing, Ohloh analysis, Software analysis, Static analysis
22	Programming concepts	Knowledge related to specific areas of the programming discipline.	Button, Concurrency, Error handling, Files handling, Graphics design, Graphics programming, Image processing, Menu, Modal dialog, Object Orientation, Parser, Privacy, Reactive, Shortcuts, String handling, Threads, Tooltip, UI coding, Unicode, Usability, Video programming, Web programming
10	Programming language	A formal language comprising a set of instructions that produce various kinds of output.	C, C++, HTML & CSS, Java, Javascript, PHP, Python, Vala, XML
8	Project architecture	Describe the aspects of how the project is built and implemented.	Adopted Design patterns, Architecture, Browser interaction with plugins, Project component: input data, Configuration files, External packages, Level of customer service, Project structure
5	Project specific concepts	Information related to the domain, the documentation and other artifacts used in the software.	GIMP project, GUI Facebook, Project UI, Pygame project, VLC documentation
3	Testing	Related to techniques and methods to test the software.	Diagnostics tests, Testing environment, Unit testing

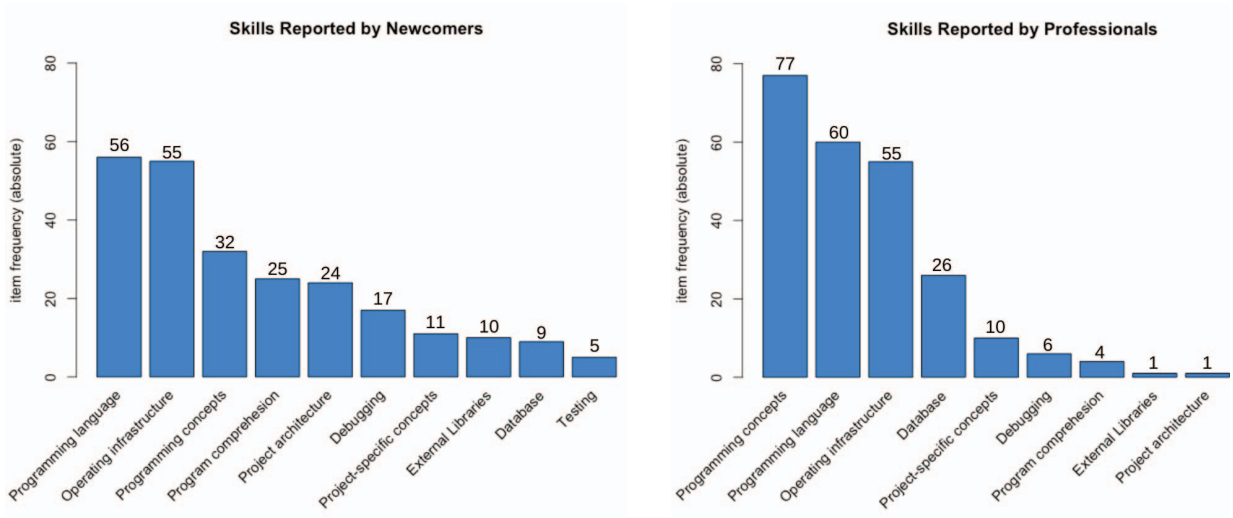

Figure 3. Frequency of skills categories (students and professionals).

P4 mentioned that "*I look at a feed of new issues, and if I see something interesting, then I look into it further.*"

Recommendations to newcomers. We use the responses to questions: *What advice do you recommend for newcomers, when they have to identify skills from GitHub issues?* and *What information should they look at?* helped collect professionals' recommendations for newcomers (see Table V). Most professionals recommend reading the issues carefully, paying particular attention to the issue title, description, and possible labels to gather any new information about the issue (mentioned by P3, P4, P5, P6). P3 said: "*read and comprehend title and description, search for familiar terms or sentences. Look at the possible tags that other members have given the issue.*"

They also recommended reading in detail the README file to understand how the project works (P3). Another professional (P5) recommend looking for issue tags used in the project: "*Search for 'good first issue' tag! That is what GitHub recommends itself.*"

Identifying skills from issues. We collected this data from participant responses to the question: *How did you identify the skills from GitHub issues?* (see Table VI). Professionals follow different approaches to identify the skills from GitHub issues, but they have some commonalities. Professionals mentioned that they would look into the title, description, comments, and

Table III
SKILLS CATEGORIES CORRECTLY IDENTIFIED BY STUDENTS

Categories	# of identified skills P	S	Correct	Precision	Recall	F-measure
Database	26	8	6	0.75	0.23	0.35
Debugging	6	24	3	0.12	0.5	0.19
External Libraries	1	10	0	0	0	0
Operating infrastructure	55	48	32	0.66	0.58	0.61
Program comprehension	4	28	1	0.03	0	0
Programming concepts	79	28	18	0.64	0.22	0.32
Programming language	64	60	34	0.56	0.53	0.54
Project architecture	1	27	0	0	0	0
Project specific concepts	11	13	0	0	0	0
Testing	0	4	0	0	0	0
Legend: P = Professionals, S = Students						

Table IV
STRATEGIES TO IDENTIFY TASK WITH PROFESSIONALS EXPERTISE

ID	Strategy
P1, P2, P3	Selected projects that use specific programming languages.
P4, P5, P6	Search for issues exciting and challenging.

labels of the issues and try to reproduce the issue to evaluate the cause of the bug and from that gauge if they have the required expertise (mentioned by P2, P3, P4, P5, P6).

Some professionals mentioned that they would look for certain keywords to identify the skills. With that information, they then attempt to identify the issue type, the time needed to solve it, and assess if they have the required knowledge. As mentioned by P3, "*I look for keywords like languages, possible components, classes, and tools that are possibly required to solve the issue. Using these keywords, I try to identify the type of the issue, how long to solve it and whether or not I can do it with my knowledge. Next, I read some documentation to verify how to contribute and install the app. Finally, doing this I install the app to verify the reproducibility of the environment and the problem in my machine.*"

Skills that are difficult to identify. Table VII collates the responses to the question: *Which issues from the categories defined in our research, do you think are more difficult to identify?* Professionals mentioned that they found it difficult to identify issues from the following categories: (i) operating infrastructures (P1, P2, P3); (ii) program comprehension (P2, P3); (iii) project specific concepts (P3, P4, P5); (iv) external libraries (P3, P4, P6); (v) programming concepts (P4); (vi) testing (P4); and (vii) project architecture (P5, P6).

Table V
ADVICE TO NEWCOMERS TO IDENTIFY ISSUES FROM ISSUES

ID	Recommendation
P1, P3, P6	Read project documentation (e.g., README file).
P2	Try to make small contributions.
P2	Be aware of your motivation for the chosen project.
P3, P4, P5, P6	Read the labels, tags, titles, and descriptions of the issues.
P5	Search for tags indicated for the newcomers (e.g., good first issue tag).

Table VI
STRATEGIES TO IDENTIFY SKILLS FROM ISSUES

ID	Recommendation
P1	Contact people experienced in the project.
P2, P3, P4, P5, P6	Read the issue (e.g., title, description, comments) to measure the impact and complexity.
P3, P5	Read the project documentation and try to reproduce the issue reported.

As observed by P2: "*Operating infrastructure and program comprehension. Both of those can be more difficult to discern from a GitHub issue due to the amount of work involved to understand what needs to be done. The rest seem to be pretty clear through reading the description of the issues*", and, as mentioned by P1 "*The most tricky problems are the ones where two Operating Infrastructures interact and both sides blame the other.*"

We observe that some of these categories of skills are the ones where students had more difficulty as well. It points to new research direction on how to make the skills from those categories easier to identify for professionals as well as newcomers.

Table VII
SKILLS CATEGORIES DIFFICULT FOR PROFESSIONALS TO IDENTIFY

ID	Recommendation
P1, P2, P3	Operating Infrastructures
P2, P3	Program comprehension
P3, P4, P5	Project specific concepts
P3, P4, P6	External Libraries
P4	Programming concepts
P4	Testing
P5, P6	Project architecture

V. DISCUSSION

Contributing to OSS requires different types of skills, where the skills could pertain to technical knowledge or knowledge about OSS processes. Technical knowledge can encompass knowledge about specific programming concepts (e.g., inheritance or event handling), language-specific constructs (e.g., Java Annotations, Java Collections), frameworks or API usage (e.g., Maven, Node.js), or project-specific concepts (e.g., code functionality, specific classes or methods). Additionally, newcomers may also need knowledge about tooling for specific development life cycles (e.g., build management or DevOps).

Our results show 10 main categories of skills needed to work on OSS issues. The categories that appeared more frequently included skills related to *programming language* and *operating infrastructure*. This is expected as newcomers need to know on how to program using the programming language used in the project. However, to make a successful contribution, it is not enough to just have programming skills. Newcomers also need to understand the different components and tools of the operating infrastructure. As reported by Steinmacher et al. [15], setting up the workspace is one of

the most recurrent barriers faced by newcomers. The initial setup of the development environment requires specific skills such as knowledge on server configuration, operating systems, hardware, and network protocols.

In the following, we discuss possible research directions that emerged from our results, related literature, and implications.

Implications for educators. Using OSS contributions as part of Software Engineering courses is becoming more common. Having students contribute to open source as part of their training is a win win for the students as well as OSS projects. Contributing to a real project gives students valuable real-life experience and allows them to add this experience to their resume. For OSS projects, students participation provides an opportunity to recruit new contributors.

However, it is difficult for newcomers to figure out whether they have the right skills to contribute to a project and even after selecting a project, it is not easy to identify what can be an appropriate issue to resolve. If there is a mismatch between the skills required to resolve an issue and that which the student possesses, it can discourage and demotivate them from making future OSS contributions.

Our results highlight that students have difficulty in identifying skills from issues in OSS projects. Students in our study were better able to identify skills related to database, operating infrastructure, programming concepts, and programming language; and had difficulty in identifying skills related to debugging and program comprehension.

This suggests that although students had been trained in software engineering, they require additional training in understanding how large, real-life projects work and how their code is structured. We posit that students might also need further training in how live projects use debugging tools. testing infrastructure, and DevOps technology, as these were skills that students had difficulty identifying.

Our study has provided a classification of different skills required to contribute to OSS. Educators can use this classification, as well as the lower-level skills identified in the paper, to verify that students are being appropriately trained in these areas to better prepare them for industry.

Educators should also consider how to better scaffold the introduction of students to OSS. The professionals in our study recommended specific guidelines on how to identify the skills required for a task, a key part of which is to reproduce an issue. A systematic guideline of how to select an issue, including different ways to reproduce an issue would help students get a deeper understanding of the issue and the project characteristics

Finally, contributing to OSS projects can be high-risk, as contributions and communication are public and persistent. Making a contribution to a large, active project provides a great experience, but can also be stressful, especially for those who have low computer self-efficacy and whose first language is not English. Educators can curate a set of OSS projects that different universities can use to train their students. These projects need to be sufficiently large, but not too complex and include programming language and software engineering technology that is being taught in universities. It is possible that universities open source their capstone projects, which can then accept contributions from more junior students, who can be mentored by the senior students.

Implications for research. Understanding how newcomers identify skills in OSS projects highlights diverse research opportunities. Our results showed which categories were students are better (e.g., database, operating infrastructure, programming concepts, and programming language) and which skills they have difficulty identifying (e.g., debugging, program comprehension). This result points to research directions on how we can improve newcomers' ability to identify skills from OSS issues and investigate what makes it difficult for students to identify the skills.

Given the difficulty in identifying skills by newcomers, future research that is able to automatically extract and label issues with the skills needed to complete a task would be useful. recent work by Santos et al. [40] takes a first step in automatically labeling the issues in an OSS project with relevant APIs (or libraries) for that issue. Further research is needed to automate the identification of the skills identified in our work.

Implications for OSS communities. Academic research as well as OSS sponsored initiatives (e.g., Up For Grabs, Mozilla, Audacity, etc.) have identified best practices to facilitate newcomer onboarding. One such practice is to gather and tag issues with labels as "first good issue" or "newcomer friendly" issue. However, simply labeling an issue with such tags doesn't guarantee that the issue matches the newcomer's skills. An issue tagged with "Newcomer friendly" label may still involve technology or concepts that a newcomer doesn't possess and can be misleading. Therefore, in addition to identifying newcomer friendly issues, contributors should also tag such issues with the skills needed to resolve them.

For a newcomer to be able to correctly identify skills needed in an issue, it is expected that the issue provides enough information. According to our results, professionals recommend that newcomers carefully read the issue details (e.g., title, description, tags) to better understand the problem reported and identify the issue complexity. Thus, OSS projects should invest in improving the project description and their contribute.md to reflect the technology used, and the skills they expect from their contributors. The projects should also ensure that the issue description contains sufficient details.

Our results describe how professionals identify skills from issues. For example, professionals reproduce the issue, which is a good practice because the attempt to reproduce the issue can lead to a deeper understanding of the issue and project characteristics. Thus, OSS projects should provide systematic guidelines on how to reproduce an issue with the project technology to help newcomers onboard. The project should also provide guidelines (to current contributors) on how to report issues, such that there is enough depth to the issue description.

Finally, previous studies related to onboarding developers in software teams have investigated how to provide customized

tasks to help new developers in their onboarding process [41]. This can be a good direction for OSS projects too, where some issues could be designed to facilitate a newcomer's first interaction with the project, providing description of the skills required to complete the task and mechanisms to reproduce the error. Such "tutorial" tasks can encourage a newcomer to easily complete their first issue helping them become familiar with the project and its technology as well as boosting confidence [2], [6].

VI. Threats to Validity

We asked students to identify the skills necessary to contribute to open source in various projects available on GitHub. From the data gathered, we cannot generalize that the skills identified represent all the skills and knowledge required to contribute to OSS.

We collected data from professionals with different backgrounds and continued assessing the issues until we had 100% of the total issues analyzed by students labeled. The professionals who participated in our study have on average more than ten years of experience with software development and had a diverse range of knowledge within the computer science spectrum (e.g., programming, management, software testing, software architecture/design, software optimization, static analysis, debugging, reverse-engineering, and databases). To mitigate this, we asked them to fill a follow-up survey to give us feedback about how they perform the skills identification and provide a deeper understanding of the task completed. Moreover, the respondents mentioned that they were confident about the accuracy of the labels provided, when we asked them about that.

We compare the number of skills reported by students against the skills reported by professionals. Some of the professionals who participated in the study have some previous knowledge about the project analyzed. In future work, we plan to compare professionals against students on a project that both of them never contributed.

Data interpretation can also lead to bias. To mitigate subjectivity, we employed two researchers independently coding the answers and then we discussed and resolved any conflict.

VII. Conclusion

OSS projects offer students real software engineering settings to learn skills. Among their characteristics, OSS projects are available online and foster open contributions from individuals who want to learn and be part of a software community. Correctly identifying the right skills to solve an issue can help newcomers thrive in their first attempt to contribute to an OSS project. However, newcomers need additional information about the tasks or support from the community to identify a task suited to their skills. Research thus far has not focused on models that articulate the skills necessary to contribute to an OSS project or on how to model skill acquisition trajectories. Therefore, in this study, we investigate what student newcomers consider as skills and how their perception matches with those of software engineering professionals.

Based on our analysis, we identified a set of 94 skills that were classified into 10 higher-level categories. This skill categorization can help other studies organize skills in GitHub issues. Among the issues analyzed, the categories that had more mentions were programming languages and operating infrastructure.

Our results highlight that student performance is not optimal in identifying skills from issues in our dataset of 47 OSS projects. We also identified the skill categories where students performed better (e.g., database, operating infrastructure, programming concepts, and programming language) and worse (e.g., debugging and program comprehension).

In future work, we intend to conduct observational studies to better understand how professionals identify issues to work on. Such an understanding will help us create a systematic process to guide newcomers and even experienced developers in identifying skills correctly. Further, we plan to use the results from this study to build tools that will help in automatically tagging issues with more accurate information about the skills needed to resolve the issue [6]. Such skill labeling will help newcomers identify issues that are more suited to their current knowledge, aiding their work and bringing more newcomers to contribute to OSS projects.

Acknowledgment

This work is partially supported by the National Science Foundation under Grant numbers 1815486, 1815503, 2008089, 1900903, and 1901031, CNPq grant #313067/2020-1. We also thank the students and professionals for their participation in our study.

References

[1] J. O. Silva, I. Wiese, D. M. German, C. Treude, M. A. Gerosa, and I. Steinmacher, "Google summer of code: Student motivations and contributions," *Journal of Systems and Software*, vol. 162, p. 110487, 2020.

[2] G. Pinto, C. Ferreira, C. Souza, I. Steinmacher, and P. Meirelles, "Training software engineers using open-source software: the students' perspective," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 2019, pp. 147–157.

[3] G. H. L. Pinto, F. Figueira Filho, I. Steinmacher, and M. A. Gerosa, "Training software engineers using open-source software: the professors' perspective," in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2017, pp. 117–121.

[4] C. Ferreira, C. Souza, G. Pinto, I. Steinmacher, and P. Meirelles, "When students become contributors: leveraging oss contributions in software engineering courses," in *Proceedings of the XXXII Brazilian Symposium on Software Engineering*, 2018, pp. 260–269.

[5] G. Von Krogh and E. Von Hippel, "Special issue on open source software development," 2003.

[6] A. Sarma, M. A. Gerosa, I. Steinmacher, and R. Leano, "Training the future workforce through task curation in an oss ecosystem," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 932–935.

[7] G. Braught, J. Maccormick, J. Bowring, Q. Burke, B. Cutler, D. Goldschmidt, M. Krishnamoorthy, W. Turner, S. Huss-Lederman, B. Mackellar *et al.*, "A multi-institutional perspective on h/foss projects in the computing curriculum," *ACM Transactions on Computing Education (TOCE)*, vol. 18, no. 2, pp. 1–31, 2018.

[8] B. Morgan and C. Jensen, "Lessons learned from teaching open source software development," in *IFIP International Conference on Open Source Systems*. Springer, 2014.

[9] D. M. Nascimento, K. Cox, T. Almeida, W. Sampaio, R. A. Bittencourt, R. Souza, and C. Chavez, "Using open source projects in software engineering education: A systematic mapping study," in *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2013, pp. 1837–1843.

[10] Y. Cai and D. Zhu, "Reputation in an open source software community: Antecedents and impacts," *Decision Support Systems*, vol. 91, pp. 103–112, 2016.

[11] D. Riehle, "How open source is changing the software developer's career." *Computer*, vol. 48, no. 5, pp. 51–57, 2015.

[12] E. Parra, S. Haiduc, and R. James, "Making a difference: An overview of humanitarian free open source systems," in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2016, pp. 731–733.

[13] G. J. Greene and B. Fischer, "Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 804–809.

[14] A. Capiluppi, A. Serebrenik, and L. Singer, "Assessing technical candidates on the social web," *IEEE software*, vol. 30, no. 1, pp. 45–51, 2012.

[15] I. Steinmacher, T. Conte, M. A. Gerosa, and D. 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*, 2015, pp. 1379–1392.

[16] I. Steinmacher, T. U. Conte, and M. A. 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*. IEEE, 2015, pp. 5299–5308.

[17] Y. Park and C. 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*. IEEE, 2009, pp. 3–10.

[18] S. Balali, U. Annamalai, H. S. Padala, B. Trinkenreich, M. A. Gerosa, I. Steinmacher, and A. Sarma, "Recommending tasks to newcomers in oss projects: How do mentors handle it?" in *Proceedings of the 16th International Symposium on Open Collaboration*, 2020, pp. 1–14.

[19] J. Wang and A. 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*, 2011, pp. 76–79.

[20] T. M. Smith, R. McCartney, S. S. Gokhale, and L. C. Kaczmarczyk, "Selecting open source software projects to teach software engineering," in *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 397–402.

[21] I. Steinmacher, M. A. Gerosa, and D. Redmiles, "Attracting, onboarding, and retaining newcomer developers in open source software projects," in *Workshop on Global Software Development in a CSCW Perspective*, 2014.

[22] K. Fogel, *Producing open source software: How to run a successful free software project*. " O'Reilly Media, Inc.", 2005.

[23] I. Steinmacher, T. U. Conte, C. Treude, and M. A. Gerosa, "Overcoming open source project entry barriers with a portal for newcomers," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 273–284.

[24] I. Qureshi and Y. Fang, "Socialization in open source software projects: A growth mixture modeling approach," *Organizational Research Methods*, vol. 14, no. 1, pp. 208–238, 2011.

[25] B. Dagenais, H. Ossher, R. K. Bellamy, M. P. Robillard, and J. P. De Vries, "Moving into a new software project landscape," in *Pro-*

[28] V. Wolff-Marting, C. Hannebauer, and V. Gruhn, "Patterns for tearing down contribution barriers to floss projects," in *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*. IEEE, 2013.

ceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, 2010, pp. 275–284.

[26] G. Canfora, M. Di Penta, R. Oliveto, and S. 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*, 2012.

[27] I. Steinmacher, I. Wiese, A. P. Chaves, and M. A. Gerosa, "Why do newcomers abandon open source software projects?" in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2013, pp. 25–32.

[29] I. Steinmacher, I. S. Wiese, and M. A. Gerosa, "Recommending mentors to software project newcomers," in *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*. IEEE, 2012, pp. 63–67.

[30] I. Steinmacher, S. Balali, B. Trinkenreich, M. Guizani, D. Izquierdo-Cortazar, G. G. Cuevas Zambrano, M. A. Gerosa, and A. Sarma, "Being a mentor in open source projects," *Journal of Internet Services and Applications*, vol. 12, no. 1, pp. 1–33, 2021.

[31] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 20, no. 3, pp. 1–35, 2011.

[32] C. Macdonald and I. 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*, 2006, pp. 387–396.

[33] B. Ashok, J. Joy, H. Liang, S. K. Rajamani, G. Srinivasa, and V. Vangala, "Debugadvisor: A recommender system for debugging," in *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 2009, pp. 373–382.

[34] C. Costa, J. Figueiredo, J. F. Pimentel, A. Sarma, and L. Murta, "Recommending participants for collaborative merge sessions," *IEEE Trans. Software Eng.*, vol. 47, no. 6, pp. 1198–1210, 2021.

[35] D. Cubranic, G. C. Murphy, J. Singer, and K. S. Booth, "Hipikat: A project memory for software development," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 446–465, 2005.

[36] J. Silva, I. Wiese, D. M. German, C. Treude, M. A. Gerosa, and I. Steinmacher, "A theory of the engagement in open source projects via summer of code programs," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 421–431. [Online]. Available: https://doi.org/10.1145/3368089.3409724

[37] R. Hoda, J. Noble, and S. Marshall, "Developing a grounded theory to explain the practices of self-organizing agile teams," *Empirical Software Engineering*, 2012.

[38] I. S. Wiese, J. T. Da Silva, I. Steinmacher, C. Treude, and M. A. Gerosa, "Who is who in the mailing list? comparing six disambiguation heuristics to identify multiple addresses of a participant," in *2016 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2016, pp. 345–355.

[39] F. Herrera, F. Charte, A. J. Rivera, and M. J. del Jesus, *Multilabel Classification: Problem Analysis, Metrics and Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[40] F. Santos, I. Wiese, B. Trinkenreich, I. Steinmacher, A. Sarma, and M. A. Gerosa, "Can i solve it? identifying apis required to complete oss tasks," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 2021, pp. 346–257.

[41] A. Ju, H. Sajnani, S. Kelly, and K. Herzig, "A case study of onboarding in software teams: Tasks and strategies," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 613–623.