Barriers and Self-Efficacy: A Large-Scale Study on the Impact of OSS Courses on Student Perceptions

Larissa Salerno
lsalernodeca@student.unimelb.edu.au
The University of Melbourne
Melbourne, Victoria, Australia

Simone de França Tonhão
siimone.franca@gmail.com
State University of Maringá
Maringá, Paraná, Brazil

Igor Steinmacher
igor.steinmacher@nau.edu
Northern Arizona University
Flagstaff, Arizona, United States

Christoph Treude
christoph.treude@unimelb.edu.au
The University of Melbourne
Melbourne, Victoria, Australia

ABSTRACT

Open source software (OSS) development offers a unique opportunity for students in Software Engineering to experience and participate in large-scale software development, however, the impact of such courses on students' self-efficacy and the challenges faced by students are not well understood. This paper aims to address this gap by analyzing data from multiple instances of OSS development courses at universities in different countries and reporting on how students' self-efficacy changed as a result of taking the course, as well as the barriers and challenges faced by students.

CCS CONCEPTS

• **Applied computing** → **Collaborative learning**; • **Software and its engineering** → **Open source model**; • **Information systems** → **Open source software**.

KEYWORDS

open source software, barriers, self-efficacy, education

ACM Reference Format:
Larissa Salerno, Simone de França Tonhão, Igor Steinmacher, and Christoph Treude. 2023. Barriers and Self-Efficacy: A Large-Scale Study on the Impact of OSS Courses on Student Perceptions. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023), July 8–12, 2023, Turku, Finland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3587102.3588789

1 INTRODUCTION AND MOTIVATION

As part of their coursework, students in Software Engineering often do not get the opportunity to participate in large-scale software projects with hundreds of developers, thousands of files, and long project history. Yet, many of the challenges inherent in software development only become apparent when software development is conducted at such a large scale. While it is often unrealistic to embed students in an industry project for a semester, open source software (OSS) development offers a unique opportunity for students to experience and participate in large-scale software development.

Recognizing this opportunity, several universities are now offering dedicated courses that introduce students to OSS development and guide them in making their first contribution to an open source project. But what is the impact of such courses?

Using data from four instances of three courses at three universities in three countries and a total of 359 students, we report on how students' self-efficacy changed as a result of taking a course on OSS development, what barriers the students expected before starting, and what challenges they actually faced in retrospect.

2 RELATED WORK

Several recent efforts studied how OSS projects are used in the context of a classroom [4, 5, 8, 10, 11]. Some aimed to understand how projects used in the classroom are chosen. For example, Smith et al. [11] focused on selecting the most appropriate projects for students' work. Morgan and Jensen [8] detailed the experience of teaching a Software Engineering course based on OSS projects.

Other papers report experiences of using OSS in different courses and contexts. Buchta et al. [4] reported their experience in teaching software maintenance and evolution aspects in a Software Engineering course. Holmes et al. [6, 7] reported the lessons of their Undergraduate Capstone OSS Projects (UCOSP) in two instances. They present details of the course, benefits, and potential challenges.

Holmes et al. [6] also analyzed how students perceived the opportunity of taking the capstone course based on OSS. They report that students took advantage of the opportunity to apply their skills in real tasks, from real projects, while receiving real feedback from project maintainers. Steinmacher et al. [12] also report the perception of students who contributed to OSS projects as part of an undergraduate course. They were interested in understanding the impact of a portal on the students' perceived self-efficacy. Similarly, Pinto et al. [9] investigated the perspective of students a few years after they took a course based on contributions to OSS projects. They found that students recurrently report challenges from social, process, and technical natures but they also report benefits related to improving their technical skills and their self-confidence.

Differently from the previous literature, in this paper we take a closer look at how an OSS course may change the perception of students about contributing to an OSS project. We aim to understand

ITiCSE 2023, July 8–12, 2023, Turku, Finland

ACM ISBN 979-8-4007-0138-2/23/07…$15.00
https://doi.org/10.1145/3587102.3588789

the shift in terms of self-perceived efficacy and in terms of barriers expected and actually faced during the contribution process.

3 COURSE DESIGN

The courses considered in this study were Software Engineering courses with a focus on software processes; the courses had been taught by two of the authors at three institutions in three different countries, but all followed a similar structure. In class, students learned about OSS development practices, tools, processes, the history of OSS development, licenses, and research on newcomer onboarding and mining software repositories. These were taught through lectures and exercises on topics such as source code management, code review, and continuous integration using GitHub.

For assessment, students were required to complete online quizzes on the theoretical lecture material and individual "mininetnographies" in which they analyzed the progression of selected GitHub users from their first contribution to their current role in OSS development. This was intended to introduce students to role models in the field. The majority of the assessment was focused on team projects, in which students worked in groups of approximately five to make a contribution to an open source project. The lecturers provided a selection of projects for the student teams to choose from and contacted the project maintainers in advance to ensure that student contributions would be received in a timely and respectful manner. In most cases, the lecturers already had an ongoing relationship with these maintainers from previous collaborations.

Each student team was tasked with selecting a non-trivial open issue from an open source project and developing a plan for addressing it. The lecturers provided feedback on these plans through a short team presentation to guide the teams as needed. The teams then had a few weeks to complete their proposed open source contribution and were encouraged to submit an initial pull request early to allow for multiple rounds of feedback from the project maintainers. The final assessment was based on team presentations and the submitted pull requests. Most of the marks were awarded based on how well each team followed open source contribution processes and interacted with the project maintainers. Whether the pull request was successfully merged played a secondary role in the grading, as it can depend on factors outside of students' control.

4 RESEARCH METHODS

In this section, we detail data collection and analysis.

4.1 Data Collection

Table 1 presents the open and close-ended questions used to collect data from the 359 students who took one of the open source courses taught at one of the three institutions considered in this study.

First, all students from all courses were asked to answer an open-ended question about the challenges that they anticipated facing **before** they began their team projects. The same question was asked **after** their contribution attempt, to provide an account of the challenges they actually encountered.

Using the same **before-after** design, we administered a self-efficacy questionnaire with a five-point Likert-scale (Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree) to measure the impact of the course. Self-efficacy is a measure of the confidence in the participants' perceived ability to perform a task, which can impact one's actual ability to complete a task [2]. The questionnaire applied was borrowed from Steinmacher et al. [12]. The items had been further classified into Social, Process, and Technical categories [14], to enable a better understanding of these dimensions in the perceived experience.

4.2 Quantitative Data Analysis

To analyze the impact of the OSS course on the students' self-efficacy, we first mapped the Likert scale answers to an ordinal (numeric) scale, from 1 to 5 (with 1 representing Strongly Disagree and 5, Strongly Agree). We kept only the entries from those students who provided answers both **before** and **after** the course (n=359).

We, then, mapped a mixed-effects logistic regression model. We used the *answer* provided per question as our dependent variable and the item *type* (social, process, or technical) and *when* the answer was provided (**before** or **after** the course) as fixed effects. We also modeled the *participant* and the *item* itself as random effects.

We used ANOVA to evaluate the differences between **before** and **after** answers per type. We used the estimated marginal means (EMMs) to compare among groups after fitting the model and reporting the effect size per item type.

4.3 Qualitative Data Analysis

We analyzed the open-ended answers using the Thematic analysis [3] approach. Thematic analysis is a method of analyzing qualitative data that involves breaking down and coding the text into meaningful themes. It is useful for understanding how different pieces of information are related, identifying patterns in responses, and uncovering underlying meanings within a dataset. Thematic analysis provides researchers with an effective way to explore their data more deeply by allowing them to identify key ideas or concepts that can be used as the basis for further research or decision-making.

The analysis consisted of two phases, one phase to analyze answers to the question related to challenges the students anticipated **before** contributing, and another phase for the question about the challenges that students faced **after** contributing to an OSS project. Two of the authors worked individually on thematically analyzing the challenges reported by the students. The researchers analyzed each answer and derived themes according to the content of each student's response. For simplicity's sake, we decided to adopt the same themes and categories nomenclature from two similar studies that one of the authors conducted in the past [1, 13]; some of the categories did not have a correspondent, so for those we added new terms to the original nomenclature. The outcome of the analysis was a list of themes that were placed into five categories: Newcomers' orientation, Newcomers' characteristics, Communication, Documentation problems, and Technical hurdles.

In total, 265 students responded the **before** questionnaire, and 191 students answered the **after** questionnaire. We analyzed the data combining the three courses. The themes were generated according to the content of the students' answers. The researchers read each response carefully, derived a list of challenges reported in each answer, and then generated themes by merging challenges. For example, one participant mentioned – *"Learning the syntax and language that the project uses might take awhile depending on the*

Table 1: Survey Questions

Before		
What challenges do you expect to encounter when trying to make a source code contribution to an open source project?	Open-Ended	
After		
What challenges did you encounter when trying to make a source code contribution to an open source project?	Open-Ended	
Before and After		
I feel comfortable asking for help from the open source community using electronic communication means.	Likert-Scale	
I can write my questions and understand answers in English.	Likert scale	Social
I am good at understanding code written by other people.	Likert scale	Social
I feel comfortable with the process of contributing to an open source project.	Likert scale	Process
I think that contributing to an open source software project is an interesting activity.	Likert scale	Process
I feel I can set up and run an application if a set of instructions is properly given.	Likert scale	Process
I can choose an adequate task to fix if a list of tasks is given.	Likert scale	Process
I am pretty good at searching for solutions and understanding technical issues by myself.	Likert scale	Technical
I have pretty good skills to write and change code.	Likert scale	Technical
I can find the piece of code that needs to be fixed given a bug report presenting the issue.	Likert scale	Technical

language and application towards that project". From that chunk of text, we identified that the need to learn a new programming language might be a challenge when contributing to an OSS project for the first time, so we added "Learn a new programming language" to the list of potential themes. In general, the two researchers had no difficulty in reaching a consensus, as the themes each researcher identified were similar.

5 IMPACT ON SELF-EFFICACY

The distribution of answers **before** and **after** look very similar when we observe the boxplots in Figure 1. However, it is possible to observe a small shift in the mean of the answers after the course. Since visual inspection did not highlight any clear pattern, and since the scale used was small, we focus on the results of the logistic model to understand if there were any trends.

First, as shown in Table 2, the result of the regression showed that the fixed-effects explain more than 50% of the answers provided by the participants. Residual is greater than the variance explained by the random effects, although the individual preferences (participants) are non-negligible. Analyzing the result of the ANOVA test, we observed that the F-value indicates significant differences for both the time (before and after) and the type of the items (Social, Process, and Technical) – F-values=47.723 and 5.785, respectively.

Table 2: Random Effects analysis

Group Name	Variance	Std.Dev.
participant (Intercept)	0.25536	0.5053
item (Intercept)	0.02632	0.1622
Residual	0.51727	0.7192

Table 3: Results for the ANOVA analysis

	Sum Sq	Mean Sq	F-value
when	24.6854	24.6854	47.723
type	5.9848	2.9924	5.785

When digging deeper into the differences before and after per dimension, the result of the regression showed a significant difference when comparing the answers provided before and after the course for all three dimensions (Social, Process, and Technical) – p-value<0.001. The effect size showed an increase in the values of answers received after the course for all dimensions (≈0.16, small effect size). Given these results, we conclude that, although with a small effect size, the students perceived themselves as more confident with the OSS contribution process at the end of the courses (the small shift in the mean —Figure 1— is indicative of this improvement).

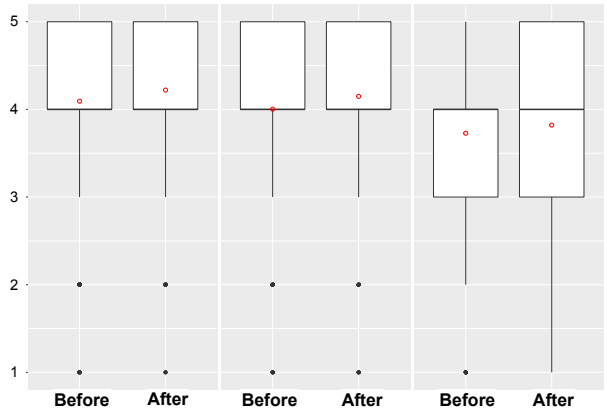

Figure 1: Distribution of answers before and after the course, with answers related to Process, Social, and Technical shown from left to right. The red circle identifies the values average.

6 BEFORE-AFTER COMPARISON

We received a total of 265 responses for the **before** contributing question and 191 answers for the **after** contributing question. The diagrams in Figure 2 and Figure 3 represent the challenges we identified through the analysis process. We used the categories names represented in similar studies [12, 13] to group the challenges. The diagram comprises five categories with multiple subcategories, representing the challenges students have reported. The approximate percentage of students who reported each challenge is displayed next to the categories and subcategories' names.

Due to the large number of subcategories that each category holds, we will focus on discussing the challenges with a higher percentage. The categories will be discussed in separate subsections, presenting the **before** and **after** results.

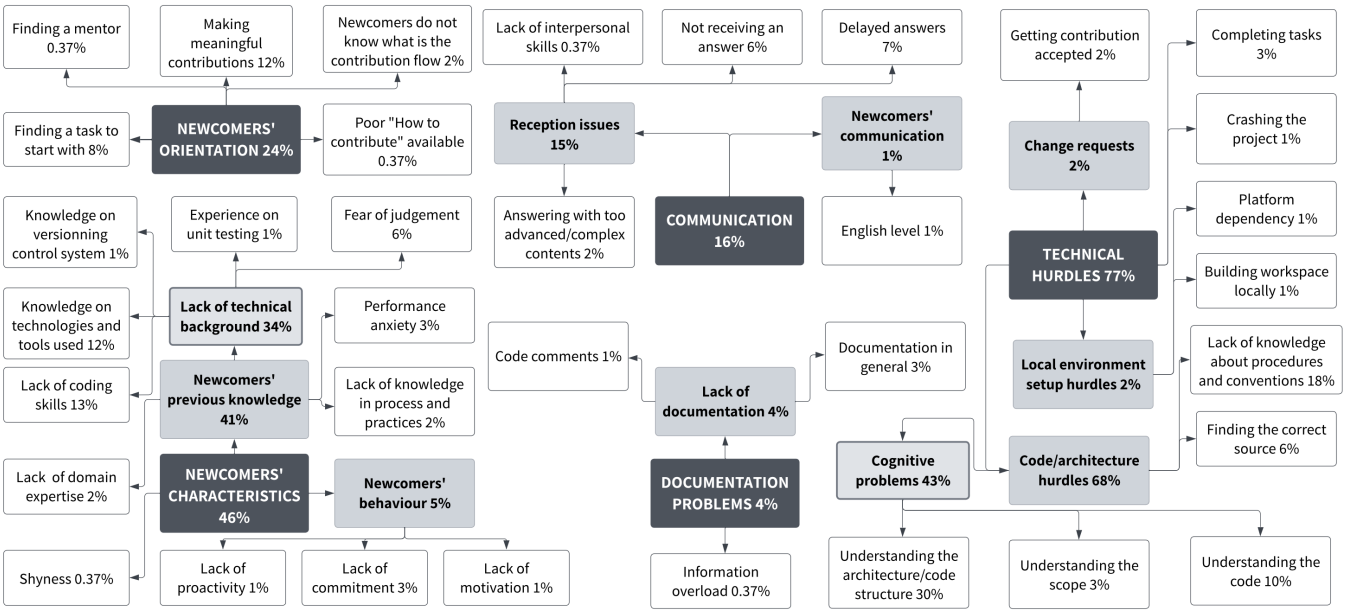

Figure 2: Challenges the students expected before contributing to an open source project.

6.1 Newcomers' Orientation

Analyzing the data, we identified a number of challenges students expected to find **before** contributing to an open source project. In total, 24% mentioned that the orientation and support they received from the community was a key factor.

Out of the five subcategories, "Making a meaningful contribution" and "Finding a task to start with" were the most prominent, with 12% of students indicating that not being able to make a meaningful contribution to the project could be a challenge. One of the participants said – *"I think the big challenge is when I have to create a new useful feature for the project. I need to define goals and objectives"*. Finding a task to start with is also an aspect that students mentioned, 8% stated that the process of picking an issue to work on could be challenging, especially because of their skill level.

After contributing to an open source project, 29% of students encountered challenges related to the orientation they received. The percentage slightly increased, but the most noticeable change is in the subcategories. Only 2% of the students stated that making meaningful contributions was challenging, but 16% mentioned that finding a task to start with was difficult. The main reason why is that they could not predict the task difficulty level. One of the participants mentioned mentioned how their team struggled to pick the right issue to work on – *"My team had trouble choosing an issue because we have no idea how difficult one issue is"*, another student mentioned – *"Identifying the "good first issue" for our team to take up. There were numerous issues on the [project] GitHub repository. We wanted to make sure we do not pickup [sic] too hard or too easy to solve issue as the contribution should be significant enough"*.

6.2 Newcomers' Characteristics

Before making a contribution to an OSS project, 46% of the students expected that their characteristic traits could potentially be a challenge. In terms of their previous knowledge, 34% of the students believe that their lack of technical background could be an issue, especially when it comes to their lack of coding skills and knowledge of technologies and tools used in the project.

The lack of technical background proved to be a significant hurdle after their contribution. Approximately 22% of the students faced challenges regarding the technologies and tools used in the projects. The need to learn new programming languages and use tools they had never used before was the main aspect reported by students – *"Before I worked on this project, I knew nothing about JavaFX, and I have not used Gradle once. It was hard for me to learn a new thing from scratch, especially JavaFX since it is a minority framework that not too many people are using it"* said the student. On the other hand, the fear of students regarding their lack of coding skills did not become a reality, as only 3% of the students reported facing challenges related to this aspect.

6.3 Communication

Regarding communication barriers, only 16% of the participants expected to face any problems in this sense; most participants were expecting to face reception issues, which included receiving delayed answers and not receiving an answer from the community. After contributing, 27% of the participants encountered communication challenges; 12% of participants mentioned that they received delayed answers and 2% did not receive any answer or feedback – *"Our final challenge was getting a response from the Project owner. Although we made a PR, we never got the feedback"*.

The participants' English level was also reported as an issue, as the majority of students are from non-English speaking countries – *"Since most of the open Source projects are English, the language is an important question. Sometimes I can't correctly get the idea about what should I do. I'm confused about my goal. It troubles me deeply"*.

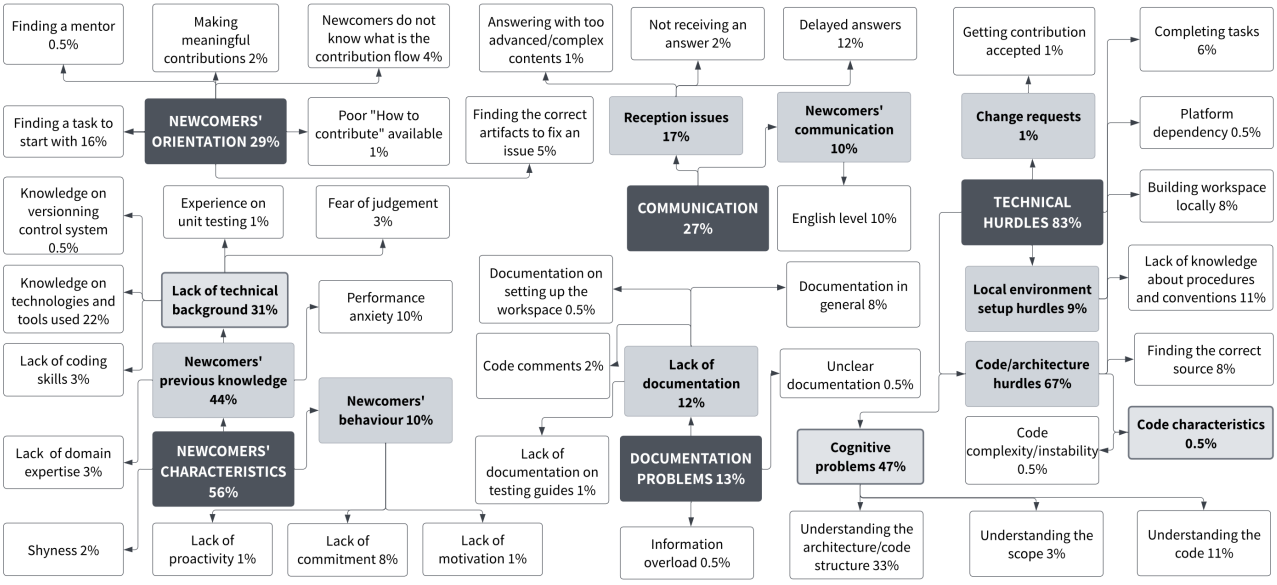

Figure 3: Challenges the students encountered after contributing to an open source project.

6.4 Documentation Problems

As the students were having contact with the projects for the first time, the documentation to understand the project and the code was crucial. **Before** contributing, only 4% of the participants were expecting to find a lack of documentation about the projects they would be working on. However, this scenario changed **after** contributing, when 12% of the participants mentioned that one of the challenges they faced was the lack of documentation about the project. The lack of documentation in general, comments in the code, testing guides, and setting up the environment documentation were aspects they indicated as challenging **after** contributing.

6.5 Technical Hurdles

The category of technical hurdles emerged as the most frequently cited by the students both **before** and **after** contributing. **Before** contributing, 77% of the students were expecting to face technical challenges, such as challenges related to the understanding of the code structure and architecture of the project.

Approximately 30% of students expected challenges when understanding the architecture of the project and code structure – *"Some source code can be very hard to read depending on the structure. The code might be separated into different files with dependencies from another file. Understand what the code is already doing can take some time before making any contribution to the code"*; another student said – *"Understanding the code structure could take quite a while due to either an unusual code style or the size of the project"*.

Besides the architecture and code structure, 10% of the students were expecting to face challenges in understanding the code itself, especially in terms of understanding different coding styles – *"The main challenge I would encounter is the steep learning curve that comes with understanding code written by other people. I find the starting point to be the most difficult in every project"*. One student also pointed out how the variety of coding styles can affect the code readability – *"As the community grows and a lot of developers participate, it will be challenging to establish coding standards, as each individual has their own style in coding. This might impact the readability of the code"*.

The lack of knowledge about procedures and conventions was also a concern for 18% of the students **before** contributing. Students believe that not meeting the projects' code standards could prevent them from having their contribution accepted – *"It's possible to receive some negative feedback from the community if my coding practice does not meet the standard"*; another student also shared the same perception – *"Another challenge will be adhering to the requirements of the submission such as some projects might have a certain code coverage that needs to be added or to ensure that all existing test cases pass"*.

The concerns students had regarding technical hurdles became a reality, as 83% of the students faced technical challenges while making contributions to the projects. Understanding the code structure and architecture was the major challenge, being mentioned by 33% of the students – *"As expected, understanding the project structure is hard. I was at a loss for how to start at the beginning"*; another student reported – *"Jumping on board and knowing nothing about the source code or the software architecture was quite challenging"*.

Understanding the code was also a challenge for 11% of the students, similarly to **before** contributing. The main issue was related to the different coding styles – *"The coding style was inconsistent across the files of the project. As a result, our team had to take more time trying to figure out which coding style was the most common"*. The lack of knowledge about procedures and conventions also happened to be an issue for 11% of the students. According to one of them, the project demanded a strict standard, but it was not offering any information or instructions – *"Abiding by coding standards/style of the open source project. We had a few issues where our pull request was not accepted due to the way we did the task"*.

7 DISCUSSION

In this section, we discuss the implications of our results for educators as well as threats to the validity of our study.

7.1 Implications

A single contribution experience is not sufficient. Our study revealed that a single OSS course might not be enough to improve students' self-efficacy and ability to overcome barriers. Our findings showed that some of the challenges students anticipated turned out to be true, while others were even more difficult to overcome than expected. For instance, we observed an increase in performance anxiety, from 3% expected to 10% encountered. To address this, we recommend educators consider incorporating OSS contributions into multiple courses and/or encouraging participation in programs such as Google Summer of Code, despite scheduling constraints.

Tools and technologies are crucial for success. Our study found that students were initially worried about their lack of coding skills (13%) and knowledge of tools and technologies (12%) **before** starting their OSS contribution journey. However, in hindsight, they realized that knowledge of tools and technologies (22%) was a much greater issue than coding skills (3%). While coding skills are necessary, we recommend educators to incorporate tools and technologies into their curriculum. Real-world software projects, both in industry and open source, rely heavily on tools and technologies, and students' proficiency in using them is crucial for their future.

Documentation issues are more prevalent than anticipated. One of the most striking differences in our "**before**" and "**after**" survey responses was related to documentation problems. While only 4% of the students expected such issues, they were actually encountered by 13%. The problem of inadequate or outdated documentation is a well-established issue in Software Engineering literature. Therefore, we urge educators to prepare students for the realities of software documentation, including teaching them how to write clear and comprehensive documentation and how to navigate code bases where documentation may be lacking.

Non-native speakers face difficulties with conventions, communication, and documentation. Large-scale software development is equally about communication and collaboration as it is about programming. English is the primary language of communication in most projects, and students for whom it is not their first language can struggle with understanding and adapting to common conventions and styles. Educators can support these students by providing them with templates for effective first messages to a project or pull request titles and descriptions. Encouraging them to study and learn from successful contributions by other open source contributors can serve as a guide for them to communicate effectively with contributors from diverse backgrounds.

Understanding code and code structure requires significant effort. This includes becoming familiar with coding styles, conventions, and best practices used in the project. However, the strict time frame of a university course does not align well with the flexible nature of OSS contributions. OSS projects are ongoing and can take a long time to fully understand, whereas university courses are often limited to a specific semester or term, creating a mismatch between the two environments. Educators should be aware of this mismatch and provide students with adequate time and resources to become proficient in navigating and contributing to OSS projects **before** they are ready to work on their first contribution.

7.2 Threats to Validity

The conclusions we make are based on data from 359 students from four instances of three courses at three universities in three countries. While we consider this to be a large sample size, we note that our findings may not necessarily generalize to other courses or student populations. Additionally, the scope of the projects that the students contributed to is relatively limited, and primarily comprises projects that the lecturers were familiar with. This may not accurately reflect the experience of students who work on projects that are not specifically advised to expect student contributions. Furthermore, the qualitative analysis component of our research introduces an element of subjectivity. To address this concern, we employed a nomenclature in line with similar studies in the field.

The survey responses may be influenced by social desirability bias, where participants provide responses that they believe are socially acceptable rather than accurate. To mitigate this threat, we emphasized to students that their answers would not affect their grades. Additionally, the measures used to assess self-efficacy may not be completely reliable or valid. We followed established practices in phrasing the self-efficacy questions in the survey.

8 CONCLUSION

This paper aimed to address the gap in understanding the impact of OSS development courses on students' self-efficacy and the challenges faced by them. Through analyzing data from multiple instances of OSS development courses at universities in different countries, we found that students' self-efficacy slightly improved as a result of taking the course. Additionally, we identified that many of the challenges anticipated by students actually occurred, with issues related to tools, technologies, and documentation being more prevalent than expected. Based on these findings, we provide implications for educators on how to best guide students to make successful contributions to an OSS project.

Future research in this area could aim to better understand the long-term effects of participating in open-source software development on students' careers and professional growth. Additionally, it would be interesting to explore how the students' Software Engineering skills were impacted by their experience with the OSS course as well as investigate the differences between the courses in more detail. Thus, it would be valuable to explore and develop effective strategies and best practices for both OSS projects and educators to support and guide students through any challenges and barriers they may encounter during their participation in OSS development. This could not only enhance students' learning experiences, but also increase the chances of successful contributions to open-source projects, ultimately promoting both students' education and the sustainability of OSS projects.

ACKNOWLEDGMENT

This work is partially supported by the National Science Foundation under Grant number 2247929.

REFERENCES

[1] 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 (April 2018), 679–714. https://doi.org/10.1007/s10606-018-9310-8

[2] Albert Bandura. 1986. The explanatory and predictive scope of self-efficacy theory. *Journal of social and clinical psychology* 4, 3 (March 1986), 359–373. https://doi.org/10.1521/jscp.1986.4.3.359

[3] Virginia Braun and Victoria Clarke. 2012. *Thematic analysis.* American Psychological Association, Washington, D.C., USA. https://doi.org/10.1037/13620-004

[4] Joseph Buchta, Maksym Petrenko, Denys Poshyvanyk, and Václav Rajlich. 2006. Teaching evolution of open-source projects in software engineering courses. In *2006 22nd IEEE International Conference on Software Maintenance* (Philadelphia, PA, USA) *(ICSM)*. IEEE, New York, NY, USA, 136–144. https://doi.org/10.1109/ICSM.2006.66

[5] David Coppit and Jennifer M Haddox-Schatz. 2005. Large team projects in software engineering courses. *ACM SIGCSE Bulletin* 37, 1 (February 2005), 137–141. https://doi.org/10.1145/1047124.1047400

[6] Reid Holmes, Meghan Allen, and Michelle Craig. 2018. Dimensions of experientialism for software engineering education. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training* (Gothenburg, Sweden) *(ICSE-SEET '18)*. ACM, New York, NY, USA, 31–39. https://doi.org/10.1145/3183377.3183380

[7] Reid Holmes, Michelle Craig, Karen Reid, and Eleni Stroulia. 2014. Lessons learned managing distributed software engineering courses. In *Companion Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE '14)*. ACM, New York, NY, USA, 321–324. https://doi.org/10.1145/2591062.2591160

[8] Becka Morgan and Carlos Jensen. 2014. Lessons learned from teaching open source software development. In *IFIP International Conference on Open Source Systems* (San José, CR) *(IFIPAICT '14)*. Springer, New York, NY, USA, 133–142. https://doi.org/10.1007/978-3-642-55128-4_18

[9] Gustavo Pinto, Clarice Ferreira, Cleice Souza, Igor Steinmacher, and Paulo Meirelles. 2019. 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* (Montreal, Canada) *(ICSE-SEET '19)*. IEEE, New York, NY, USA, 147–157. https://doi.org/10.1109/ICSE-SEET.2019.00024

[10] Anita Sarma, Marco Aurélio Gerosa, Igor Steinmacher, and Rafael Leano. 2016. 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* (Seattle, WA, USA) *(FSE'16)*. ACM, New York, NY, USA, 932–935. https://doi.org/10.1145/2950290.2983984

[11] Therese Mary Smith, Robert McCartney, Swapna S Gokhale, and Lisa C Kaczmarczyk. 2014. Selecting open source software projects to teach software engineering. In *Proceedings of the 45th ACM technical symposium on Computer science education* (Atlanta, Georgia) *(SIGCSE '14)*. ACM, New York, NY, USA, 397–402. https://doi.org/10.1145/2538862.2538932

[12] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming open source project entry barriers with a portal for newcomers. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. ACM, New York, NY, USA, 273–284. https://doi.org/10.1145/2884781.2884806

[13] Igor Steinmacher, Marco Gerosa, Tayana U Conte, and David F Redmiles. 2019. Overcoming social barriers when contributing to open source software projects. *Computer Supported Cooperative Work (CSCW)* 28, 1 (June 2019), 247–290. https://doi.org/10.1007/s10606-018-9335-z

[14] Igor Steinmacher, Christoph Treude, and Marco Aurelio Gerosa. 2018. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software* 36, 4 (January 2018), 41–49. https://doi.org/10.1109/MS.2018.110162131