What barriers do students experience when trying to contribute to Open Source Software projects?

Nathan Cassee
University of Victoria
Victoria, Canada
nathancassee@uvic.ca

Sankarsh Ravi
Eindhoven University of Technology
Eindhoven, The Netherlands
sankarsh72000@gmail.com

Italo Santos
University of Hawai'i at Mānoa
Honolulu, HI, USA
isantos3@hawaii.edu

Igor Steinmacher
Northern Arizona University
Flagstaff, AZ, USA
igor.steinmacher@nau.edu

Alexander Serebrenik
Eindhoven University of Technology
Eindhoven, The Netherlands
a.serebrenik@tue.nl

ABSTRACT

Open Source Software (OSS) supports modern digital infrastructure and industry, making the ability to contribute and collaborate in OSS communities an increasingly important professional skill. Participating in OSS gives students valuable real-world software engineering experience, but also introduces distinctive challenges. In this work, we investigate the barriers university students encounter when contributing to OSS projects. We qualitatively analyze experiences from an advanced master-level software engineering course at a European university and systematically map those experiences to existing barrier frameworks. Notably, we identify two barriers not described in prior work: "Conflicting Mentor Guidance" and "Communication Channel Ambiguity". By combining project characteristics with student reports, we also find that commit frequency is a more reliable indicator of contribution success than project size or age. Our findings show how barriers interconnect and compound for time-constrained student contributors, with interpersonal barriers often triggering or exacerbating technical and process barriers. Students who selected projects based on personal interest achieved higher success rates, underscoring the role of intrinsic motivation. We conclude with implications for practice: students should choose projects aligned with their interests and with active maintainer engagement; educators should use vitality metrics and prepare students for common barriers; and maintainers should prioritize timely communication and mentorship.

KEYWORDS

open source software, contribution barriers, student experience

ACM Reference Format:
Nathan Cassee, Sankarsh Ravi, Italo Santos, Igor Steinmacher, and Alexander Serebrenik. 2026. What barriers do students experience when trying to contribute to Open Source Software projects?. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-SEET '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3786580.3786971

1 INTRODUCTION

Open Source Software (OSS) is the backbone of the modern digital world, powering critical infrastructure, research, and virtually every software-dependent industry [17, 44]. As software pervades every aspect of society, the demand for professionals who can effectively contribute to and collaborate within OSS communities is increasingly valued. Previously, GitHub's report stated that over 5.2 billion contributions were made across public and OSS projects in a single year, underscoring the scale of such activity [14]. Industry leaders and educators now recognize fluency in OSS development as a core competency for software engineers, not an optional specialization [20, 42].

For students, engaging in OSS is more than just a résumé-building exercise; it is an entry point to real-world practices, distributed teamwork, complex codebases, and global technical communities [23, 24]. OSS assignments expose learners to tools and workflows used for software maintenance in a collaborative environment, which are essential for professional software engineers. Critically, OSS participation can foster identity, confidence, and a sense of responsibility by empowering students to become contributors rather than passive consumers of technology.

However, the reality of integrating OSS into software engineering education often falls short of this vision. Students frequently encounter technical and social barriers that are fundamentally different from those experienced in traditional academic projects [2, 36]. These obstacles include deciphering large and poorly documented codebases, navigating ambiguous community norms, and communicating across time zones and cultures. Unlike self-directed newcomers, students must overcome these challenges while managing academic deadlines and often with limited prior experience [20, 23]. As a result, the intended educational benefits of OSS assignments can give way to frustration or disengagement, affecting both student learning and the goals of OSS communities that depend on a steady influx of new contributors [34, 44].

Although there is a growing body of research on the barriers faced by OSS newcomers [2, 33], including foundational frameworks such as those of Balali et al. [2] and Steinmacher et al. [33], much less attention has been paid to how these barriers manifest themselves for students in academic settings. Existing studies have identified technical, interpersonal, personal, and process barriers, and have begun to explore the dynamics of mentoring and motivation [1, 10, 12, 13], but systematic empirical accounts focused on students remain scarce.

In this paper, we address this gap by investigating: ***RQ1: What barriers do students experience when trying to contribute to Open Source Software projects as part of their university coursework?*** Building on and extending the frameworks of Balali et al. [2] and Steinmacher et al. [33], we conduct a qualitative analysis of reflective reports produced by student teams in a master's level course on software evolution. This method systematically maps student experiences to existing barrier categories, while remaining open to new, student-specific themes and challenges. This enables us to test the applicability of established frameworks and to contribute new insights tailored to the academic context.

To further contextualize our findings, we also investigate: ***RQ2: How do project characteristics of OSS projects, student motivations for picking OSS projects, and the barriers encountered relate to student success?*** By combining student reflections, self-reported motivations, and OSS project characteristics, we identify how students can choose OSS projects to contribute to.

Our results reveal that the most common barriers faced by student groups included low response rates from maintainers, high code complexity, and insufficient documentation. Two previously undocumented barriers also emerged: conflicting mentor guidance and communication channel ambiguity. Furthermore, our findings indicate that project commit frequency was a stronger indicator of contribution success than project size or age, with higher activity levels and more responsive maintainer engagement linked to greater success (i.e., merged pull requests). Among the 28 pull requests submitted by the students, 16 were successfully merged, demonstrating that despite encountering multiple barriers, many students were able to make meaningful contributions to OSS projects.

Our contributions are as follows: (1) a detailed mapping of barriers encountered by students during OSS course assignments, grounded in and extending previous frameworks; (2) the identification of novel challenges unique to the intersection of academia and OSS; and (3) evidence-based recommendations for educators and OSS maintainers to improve the preparation, support, and retention of student contributors. Our insights can help educators with both curriculum design and community practice, supporting future OSS contributors.

2 RELATED WORK

The integration of OSS contributions into software engineering education has attracted growing attention from both educators and researchers [6, 21, 23, 24, 29]. Understanding the barriers, supports, and dynamics that shape the OSS experience of students is critical to designing effective curricula, informing community practices, and ultimately strengthening the pipeline of future contributors. In this section, we reflect on the main threads of related research, with a focus on established frameworks for analyzing newcomer barriers, mentoring and onboarding practices, educational studies of OSS integration, and work centered on student-specific experiences.

Previous works in this area cataloged the barriers faced by newcomers to contributing to OSS projects [2, 33]. Building on empirical research Steinmacher et al. [33] and Balali et al. [2] developed frameworks that describe technical, social, process, and personal barriers that hinder newcomer participation. These frameworks highlight how barriers often interact and reinforce one another, intensifying the difficulties of maintaining long-term engagement.

Other studies have examined the role of mentoring and onboarding in reducing barriers and supporting successful contributions [28]. For example, Balali et al. [1] and Tan et al. [37] have explored how mentorship, task recommendation systems, and collaborative practices can facilitate smoother entry for newcomers. Studies such as Fronchetti et al. [12] and Hannebauer et al. [15] have further investigated the diverse motivations that drive developers and students to begin contributing, while Gerosa et al. [13] have explored how these motivations evolve over time. Collectively, these works document promising strategies for scaling support and boosting engagement, while also highlighting the variability and limitations of mentoring capacity across projects, particularly for contributors who require additional scaffolding [10].

Another important line of research investigates the intersection of OSS and formal education. Studies such as those by Nascimento et al. [6, 20, 21] and Pinto et al. [23, 24] have explored the opportunities and challenges associated with integrating OSS-based assignments into university courses. Silva et al. [30] provide insight into how participation rewards and learning objectives shape student engagement in OSS-related initiatives such as Google Summer of Code. The findings of these studies indicate that authentic participation in OSS can improve student motivation, professional preparedness, and collaboration skills, but also reveal persistent difficulties in project selection, onboarding, and access to support. The need to align academic structures with community norms emerges as a recurring theme, as does the risk that unresolved barriers may undermine both learning outcomes and student confidence.

More recently, a handful of studies have begun to focus on students as a distinct group of OSS contributors. For instance, Cereceda et al. [4], Pinto and Ferreira [23], and Gerosa et al. [13] have highlighted the unique motivations, constraints, and learning trajectories of student participants, as well as the interplay between academic and OSS cultures. Hannebauer et al. [16] provided valuable insight into the mental models and cognitive dissonance experienced by newcomers, including students, as they engage with OSS projects. While these studies advance our understanding of student participation, most rely on surveys or interviews conducted after the contribution experience, with limited attention to real-time reflections or the specific ways in which established frameworks capture (or fail to capture) student challenges.

The literature offers a solid basis for understanding OSS onboarding and education. However, a notable gap still exists: we lack systematic, reflection-based accounts of the barriers faced by students during authentic OSS coursework, and it is unclear to what extent existing frameworks capture the full range of their experiences. Our study stands out from the existing literature in addressing this gap by mapping student reflections to established frameworks while remaining attentive to new student-specific themes.

3 METHOD

To investigate the barriers experienced by students during authentic OSS contribution assignments, we adopted a systematic qualitative research design centered on the analysis of reflective reports. Unlike studies that rely on post-hoc surveys or interviews, our approach draws on contemporaneous student reflections written during the contribution process. Thereby yielding richer and more immediate accounts of real challenges, and minimizing any memory bias. The methodology is anchored in established barriers for newcomers [2, 33], but is also designed to capture new student-specific themes as they emerge. Our process consists of two main components: (i) structured data collection from multiple sources and (ii) a rigorous qualitative analysis protocol. The following subsections detail the study context, data sources, ethical procedures, and analysis methods.

3.1 Research Setting

This study was conducted in the context of a master-level course in Software Evolution at Eindhoven University of Technology, The Netherlands. The students who participate in the course are all expected to possess basic programming and software engineering knowledge, including familiarity with Git-based workflows. The course is designed to bridge academic knowledge with professional practice by engaging students in authentic OSS contribution experiences. Each year, approximately 70 students participate, organized into teams of three to five. As part of a graded assignment, teams are required to select an active, engineered OSS project, identify and address suitable issues, engage with the maintainers of the project, pick two open issues, and contribute pull requests that fix these issues. The assignment emphasizes collaboration, technical rigor, and engagement with external communities.

The courses at Eindhoven University of Technology run for ten weeks, creating a very constrained time period for assignments. Therefore, this assignment has been carefully scaffolded to ensure that students can contribute meaningfully to OSS projects, thereby benefiting both students and the OSS communities with which they interact. The selection of the OSS project is governed by a structured set of criteria designed to maximize both the educational value and the feasibility. OSS projects selected by students must meet the following requirements:

- **Collaborative OSS project:** The OSS project should be an active, collaborative, and engineered software project. These requirements have been recommended previously in other, similar courses [32].
- **Language:** The primary implementation language must be Java, Python, C#, JavaScript, or TypeScript, to ensure alignment with the students' backgrounds and with the instructors' ability to understand their source code.
- **Issue suitability:** The OSS project should have open issues that can be fixed in the scope of six weeks and require the development of source code. Issues limited to documentation, translation, or similar non-code tasks were not considered acceptable.
- **Size and complexity:** The OSS project should not be too large. In every language, several large and famous projects are incredibly popular and active. To prevent disruption in these projects and increase the likelihood of maintainers responding to the students pull requests, we requested the students not to pick famous projects.

The assignment is composed of three stages. In the first stage, each team submits its project choice, along with a shortlist of GitHub issues from the chosen project that they wish to work on. The teaching assistant reviews selected projects to ensure that all criteria are met, offers feedback, and may suggest alternative options if a proposed project or issue is deemed unsuitable.

For the second stage, students were asked to fix two of the issues they selected and to open pull requests in the project. They shared the links to their pull requests with the teaching team and were encouraged to respond to the maintainer's reviews to ensure that their contributions could be merged. Note that pull request acceptance was not part of the course grading policy and that the students were aware of this.

Finally, in the third stage, two weeks after opening the PRs, students are required to submit a report in which they describe and reflect on their experiences contributing to the project they selected. Students are instructed to ground their analysis in established literature on OSS newcomer barriers, ensuring both relevance and theoretical alignment [2, 33]

3.2 Data Collection

We combine OSS contributions as described by the students in their reports with metrics collected from the projects and pull requests. This approach combines depth (student reflections) and breadth (project characteristics and objective outcomes). The three data sets we collected are described below.

1. Reflective Reports. As described previously, each student team submits a reflective report in stage three. These reports are a required component of the course, requiring students to document their rationale for project selection, describe their journey of technical and social contribution, and explicitly reflect on the barriers they encountered. Reports are typically several pages long and are written collaboratively by all team members, providing rich, contemporaneous accounts of the contribution process. For this study, we collected reports from 13 teams spanning two academic years (2022-2023 and 2023-2024), resulting in a corpus of diverse and context-rich narratives. We included only reports from teams for which all members gave their consent for use in the research.

2. Project Metadata. To contextualize the qualitative reflections, we extracted metadata of each OSS project selected by the students. This includes the programming language(s), project age and size (e.g., number of contributors, age, programming language, repository activity levels (commit frequency, issue activity), and the presence or absence of key onboarding resources such as contribution guidelines, setup documentation, and codes of conduct.

3. Pull Request Outcomes. For each team, we recorded whether their pull request was merged, rejected, or remained pending after the course. This objective measure enables us to relate student experiences and reported barriers to the outcome of their contributions and to analyze possible relationships between project characteristics, encountered barriers, and contribution success.

3.3 Data Analysis

We follow a multistage qualitative coding protocol grounded in established frameworks for OSS newcomer barriers [2, 33]. The process was designed to maximize reliability, transparency, and theoretical alignment, while remaining open to new student-specific findings. Given the size of our population, we focus on qualitative rather than quantitative analysis, and do not conduct inferential statistics.

Before coding, all five researchers participated in a calibration session, in which sample excerpts from the reports were jointly coded and discussed. This exercise ensured a shared understanding of the definitions of barriers as articulated by Balali et al. [2] and Steinmacher et al. [33], and clarified the procedure for identifying statements that could signal novel or context-specific barriers.

Three researchers independently reviewed all relevant sections of each report and highlighted passages in which students described difficulties, barriers, or frustrations. Each passage was coded using the categories defined in the framework of Balali et al. (i.e., technical, interpersonal, process, and personal) [2]. In cases where a passage described multiple barriers, multiple codes were assigned. All coding was conducted using a structured spreadsheet that tracked report sections, text excerpts, assigned codes, and coder notes. Accompanying materials, including the assignment description and project characteristics, are available as supplemental material [1].

In addition to deductive coding, researchers flagged passages that did not fit existing categories but appeared to describe significant or recurrent barriers. These potential new barriers were documented with supporting quotes and discussed in follow-up meetings.

After the initial coding phase, the researchers compared their coding assignments, discussed discrepancies, and refined the code definitions as necessary. Coding disagreements were resolved through discussion, and, when consensus was not immediately reached, a third coder served as a tiebreaker. For each barrier and each team, a final code (present, absent, unmentioned) was assigned by consensus and linked to supporting statements.

Novel barrier candidates were only added to the final taxonomy if (1) multiple coders independently flagged them, (2) they were supported by clear and representative evidence from more than one report, and (3) all coders agreed on their distinctiveness.

3.4 Ethics

All data collection and analysis procedures were reviewed and approved by the University Ethics Review Board (ERB).[2] Before accessing any data, we submitted a research protocol describing the study objectives, data sources, data handling procedures, and plans to protect the privacy of participants. After finalizing course grades, students were contacted by email and informed of research goals, data to be collected, their rights as participants, and the voluntary nature of their participation. Explicit written consent was obtained from all team members of students whose reports were included in the analysis. Reports were then anonymized by removing names, identifiers, and any potentially sensitive project or communication details. Only the approved research team had access to the original data, which were stored on secure university-hosted servers in accordance with institutional and legal privacy requirements.

4 RESULTS

The 13 student groups who consented to participate in our study submitted a total of 28 pull requests across various OSS projects (cf. Table 1). Of these, 16 were successfully merged, 6 were closed without merging, and 6 remained open at the end of the course. This distribution reveals varying levels of success in student contribution attempts, with just over half achieving integration of their work into the mainline of an OSS project.

Students selected projects spanning various domains and ecosystems, including developer tools (Certbot, Orval), visualization libraries (Matplotlib), productivity software (JabRef, Zulip), web applications (Open Library, TEAMMATES), and utility libraries (tqdm). The selected projects varied significantly in size (ranging from 12 to 1,495 contributors), maturity (established between 2003 and 2023), and commit frequency (ranging from low to high), allowing us to explore how these characteristics influenced student experiences.

Through detailed analysis and consensus-based coding by the research team, we identified multiple barriers encountered by student contributors, illustrated in Table 2. In particular, we found two previously undocumented barriers. In the following, we first discuss the newly identified barriers, then the existing barriers experienced by students, and finally, we report the relation between project characteristics and barriers.

4.1 Emerging Barriers

Our analysis identified two barriers not previously documented in the literature that impacted student contributors. These barriers represent a contribution of our work to understanding the unique challenges faced by OSS contributors.

The first new barrier, which we named "Conflicting Mentor Guidance", was reported by two groups that contributed to the TEAMMATES and JabRef OSS projects. This barrier occurs when different project mentors or maintainers provide contradictory feedback or instructions on the same issue, creating confusion about the correct approach (cf. the "Discussion of the solution: strategy" category among the reasons for confusion in code review in the study of Ebert et al. [7]). As one group noted in their report, different mentors provided conflicting guidance on implementation details, forcing students to spend additional time reconciling these viewpoints or selecting which advice to follow. This barrier highlights the challenges of distributed mentorship in OSS projects and the importance of consistent guidance for newcomers.

The second new barrier, "Communication Channel Ambiguity", was reported by the team contributing to the website calendar project, who found themselves unable to access key project communication platforms due to unclear onboarding information. This barrier effectively isolated them from informal community interactions and support channels, significantly impacting their ability to seek help and clarification. Unlike general communication difficulties documented in previous research, this barrier is related to structural access issues rather than language or cultural differences.

4.2 Identified Contribution Barriers

To analyze the barriers reported, we used the framework of Balali et al. [2] to identify the technical, interpersonal, process, and personal-related barriers experienced by the studies. Next, we describe the students' experiences through these barriers for each category.

4.2.1 Technical Barriers. Code complexity emerged as the most common technical barrier, as reported by four of the 13 groups. Groups reporting this barrier consistently struggled to understand large codebases, project architectures, and dependencies, regardless of their familiarity with the programming language. This suggests that language familiarity alone is not sufficient to overcome the technical complexities of OSS projects.

[1]https://doi.org/10.6084/m9.figshare.30229774
[2]ERB2023MCS30

Table 1: Pull requests submitted by the students.

Group	Project	Number of Contributors	Commit Frequency	Pull Request Status	Response from Developers
A	Certbot	466	High	merged merged	received received
B	VSCode-Pets	84	Low	open open open	received received received
C	Matplotlib	1495	High	merged open	received received
D	tqdm	115	Low	open open	not received not received
E	Zulip	984	High	closed closed	received received
F	website-calender	28	Inactive	closed closed	received received
G	TEAMMATES	651	Low	merged merged	received received
H	JabRef	684	High	closed merged	received received
I	Orval	170	High	closed merged merged	received received received
J	Open Library	370	High	closed merged	received received
K	Abby	12	Low	merged merged	received received
L	TEAMMATES	651	Low	merged merged	received received
M	Weblate	1205	High	merged merged	received received

Table 2: Barriers encountered by student groups.

Barrier	Type	Count	Groups
Low response rate	Interpersonal	8	B, C, D, E, F, L, M, J
High code complexity	Technical	4	C, D, E, M
Lack of documentation	Process	4	A, D, E, I
Difficulty in setting up development environment	Technical	2	I, M
Lack of newcomers background knowledge	Technical	2	C, I
Problem with process of submitting code	Process	2	B, K
Difficulty in time management	Personal	2	B, J
Task too complex for newcomers	Technical	2	E, K
Different opinions of different mentors	Interpersonal	2	G, H
Long project processes	Process	2	A, K
Lack of knowledge about procedures	Process	2	B, E
Shyness to ask questions	Personal	1	C
Difficulty in learning tools/technology	Technical	1	C
Difficulty in choosing newcomer friendly project	Process	1	D
Harsh project atmosphere	Interpersonal	1	F
Lack of English language skills	Interpersonal	1	F
Lack of mentor's interpersonal skills	Interpersonal	1	F
Communication channels problem	Interpersonal	1	F
Communication issues (timezone/place)	Interpersonal	1	J
Challenges in communicating effectively	Interpersonal	1	M

Challenges related to setting up development environments affected two groups, particularly for projects with complex dependencies or specialized configurations. These setup challenges often consumed disproportionate amounts of students' time, reducing their capacity to actually contribute code to these projects.

Two groups reported limited background knowledge of project-specific tools and technologies as a technical barrier, reflecting domain-specific gaps rather than general programming deficiencies. For instance, students contributing to specialized projects like Orval (OpenAPI/TypeScript) or Zulip (team collaboration platform) struggled to understand the domain-specific concepts and technologies required to work on these projects.

The prevalence of technical barriers indicates that while technical challenges are common, they do not necessarily predict the outcomes of contributions. This finding suggests that other factors, such as interpersonal and process-related considerations, can also play a role in determining contribution success.

4.2.2 *Interpersonal Barriers.*

Low response rates from project maintainers or community members constituted the barrier most frequently reported, affecting eight of the 13 groups. The impact of this barrier varied between projects, and groups contributing to high-frequency commit projects generally reported faster and more consistent responses than those working with less active projects. Furthermore, this barrier also affected the contribution results, as groups that experienced low response rates showed significantly lower success in combining their pull requests. Of the 17 pull requests submitted by the eight groups that encountered this barrier, only 7 were merged, while 10 remain unmerged.

Communication challenges related to time zones, English language skills, and cultural differences affected three groups. These challenges manifested differently across projects, with some groups experiencing delays in feedback cycles due to the geographic distribution of maintainers, while others encountered difficulties in interpreting communication styles or expectations. From the student reports, it appears that the impact of these barriers was more pronounced for groups working on projects without clearly established communication channels or processes.

Only one group reported experiencing a harsh project atmosphere, indicating that most OSS communities contacted by students maintain relatively welcoming environments for newcomers.

4.2.3 Process Barriers. Documentation gaps affected four groups, although the nature of these gaps varied considerably. One group reported missing and outdated setup instructions, while the other encountered ambiguities in the contribution guidelines and architectural documentation. The third group stated that they could not find documentation for testing the feature they had implemented. Groups that contributed to larger projects(having more than 100 contributors) reported significantly better documentation practices.

OSS projects with a long process to submit and get pull request approval, together with problems with the code submission process were reported by four groups. While long project processes serve important quality control functions, they create challenges for students working within the time constraints of an academic course. Groups contributing to projects with formal governance structures (e.g., organization-owned repositories, established contribution guidelines) reported higher process delays. Groups that encountered this barrier worked on popular projects, each with over 20,000 stars on GitHub.

4.2.4 Personal Barriers. Time management challenges were reported by two groups. Students struggled to balance OSS contributions with other academic commitments, especially when unexpected technical or process issues stretched the effort beyond what they had planned. This barrier appeared across projects of different sizes, domains, and complexities, indicating it stems more from the academic context than from project-specific factors.

One group explicitly mentioned shyness when asking questions as a barrier to participation. They mentioned that they were concerned about how project managers would perceive their image if the students expressed their doubts (cf. the observation of Rodeghero [26] that less experienced developers tended to feel uncomfortable interrupting a more experienced colleague for help). An interesting perspective on why many groups did not report this barrier is that it may indicate that many projects provide sufficient documentation and asynchronous communication channels, which reduce the perceived social cost of asking questions.

4.3 Relationships Between Barriers, Motivation, Project Characteristics, and Student Success

In addition to identifying the barriers reported by students in their reports, we also relate their motivation for picking a project, the characteristics of the project, the barriers experienced, and the outcomes of the students' PRs.

When comparing groups with successfully merged pull requests to those with unmerged or closed contributions, we observed several patterns associated with successful contributions. Groups with merged contributions reported fewer interpersonal barriers, highlighting the importance of community interactions and the benefits of mentor feedback in helping the student groups successfully contribute. The quality and timeliness of maintainer feedback emerged as particularly influential factors, and groups with merged pull requests reported constructive and timely responses more frequently.

Meanwhile, groups with unmerged contributions reported more process barriers (of the 12 total pull requests from the five groups that experienced process barriers, only four pull requests were merged), suggesting that clear contribution guidelines help newcomers successfully contribute. Projects with formalized contribution structures, including contributing guides, pull request templates, and issue templates, demonstrated higher pull request acceptance rates, reinforcing the importance of structured onboarding.

Although many groups reported technical barriers, these were not related with team success, suggesting that well-functioning community processes and interactions can help student groups overcome technical complexity, enabling newcomers to contribute successfully despite the presence of technical barriers. This insight highlights the need for OSS projects to prioritize community and process improvements alongside technical documentation.

With respect to **motivation**, we find a relationship between students' motivations for project selection and the barriers they subsequently encountered (cf. Table 3). Students most often selected projects based on familiarity with the programming language and availability of "good first issue" tasks (six groups), documentation quality (five groups), and the level of activity of the project (five groups). *Groups that prioritized programming language familiarity* generally encountered fewer barriers related to basic code comprehension, but still reported struggling with architectural complexity. Suggesting that, while language familiarity provides a foundation for contribution, it does not necessarily prepare students for navigating complex project architectures or domain-specific implementations. *Groups that selected OSS projects based on documentation quality* reported lower rates of process-related barriers, suggesting that well-documented projects not only facilitate technical understanding but also clarify contribution processes. However, this does not mean that the students who chose these projects did not face technical and interpersonal challenges. This indicates that documentation alone cannot eliminate all barriers to contribution. Perhaps most notably, *groups that explicitly mentioned selecting projects based on personal interest or project purpose* demonstrated higher rates of successfully merged contributions. Among the four groups that selected projects based on personal interest, a total of nine pull requests were submitted, and eight of them were merged. This finding suggests that intrinsic motivation plays a crucial role in helping students persevere through contribution challenges and ultimately achieve successful outcomes. The motivational aspect of project selection appears to be an underexplored factor in OSS contribution success, particularly for students who are required to work on an OSS project as part of academic or course requirements.

With respect to **Project Characteristics**, we find important links between characteristics, barriers, and student success.

Recall that students were instructed to carefully consider project size, warning them not to pick projects that were too small or too large. Advice stemming from existing literature [32]. However, based on our findings, project size might not be the best signal for students to select projects. Instead, the commit frequency, or activity, of the projects appears to relate more to student success. For instance, the VSCode-Pets project (84 contributors, low commit frequency), TEAMMATES (651 contributors, low commit frequency) and tqdm (115 contributors, low commit frequency) all demonstrated low responsiveness, while projects like Matplotlib (1,495

Table 3: Motivations for selecting OSS projects.

Motivation	Count	Groups
Familiar programming language/technology stack	6	A, D, E, F, H, I
Availability of "good first issue" tasks	6	A, C, G, H, J, L
Well-maintained comprehensive documentation	5	A, C, E, H, M
Active community responsive maintainers	5	C, E, I, K, M
Personal relevance project usage	4	B, G, I, M
Manageable project size complexity	3	D, F, M
Project popularity impact	3	B, E, H
Welcoming contribution policy	2	B, E
Recent Commit History	2	D, J
Opportunity for meaningful contributions	2	H, M
Clear development procedures	1	G
Educational relevance	1	G

contributors, high commit frequency), Orval (170 contributors, high commit frequency) and Open Library (370 contributors, high commit frequency) provided timely feedback. These results suggest that students should be encouraged to look for signs of activity, such as commit frequency, rather than focusing solely on project size.

Secondly, projects with high commit frequencies demonstrated not only better responsiveness to student contributions but also more comprehensive documentation, making it less likely for students to report barriers.

Furthermore, our analysis suggests that certain barriers act as "gatekeepers" that significantly influence whether students can overcome subsequent challenges. For instance, students who encountered significant response delays often found it difficult to resolve technical issues or clarify process requirements, regardless of their technical preparedness or project knowledge. Co-occurring barriers highlight the importance of identifying and addressing issues that may prevent newcomers from overcoming subsequent challenges, as this prevents students from contributing effectively to the projects of their choosing.

5 DISCUSSION

We investigated the barriers encountered by 13 teams and analyzed their experiences after contributing to OSS. Our analysis reveals key findings and implications relevant to OSS project maintainers and educators who let students contribute to OSS as part of courses.

Interconnected Barriers. Most importantly, our analysis reveals insights about how various problems in OSS projects result in interconnected barriers that combine to affect student contributors. While previous research has typically examined barriers as single challenges [2, 33], our findings demonstrate how a combination of barriers can prevent students from contributing to OSS. For instance, when maintainers exhibited low response rates (an interpersonal barrier), students failed to resolve technical setup issues or clarify contribution requirements, making these students experience both technical and process-related barriers as well.

This interconnectedness is consistent with the observations of Mendez et al. [18] that social barriers often precede and influence technical ones in collaborative software development contexts, and with the observation of Palomba et al. [22] that (interpersonal) community smells influence the intensity of (technical) code smells. Similarly, Steinmacher et al. [34] found that newcomers who encounter multiple barriers simultaneously are more likely to abandon contribution attempts than those facing isolated challenges.

The interconnected nature of the problems that create barriers appears to be particularly impactful for student contributors operating within academic timeframes. Steinmacher et al. [33] identified time constraints as a factor that affects newcomer contributions in general, and our observations suggest that these constraints may exacerbate the impact of barriers for students. For example, student groups that experienced low response rates showed a markedly lower success rate in achieving merged pull requests compared to those who received timely feedback. Of the 17 pull requests submitted by the eight groups reporting low response rates, only seven were merged while 10 remained unmerged, highlighting the impact of communication delays within academic timeframes.

Our findings suggest that OSS projects that are interested in attracting student contributors should facilitate direct communication channels with maintainers. Projects utilizing platforms such as Discord, Slack, or other online chat systems demonstrated better success in our study, as these channels allowed students to receive timely guidance from the OSS community when faced with challenges. This approach could help break the chain of problems that leads from delayed responses to unresolved technical or process-related issues and, ultimately, to unsuccessful contributions.

Project Health as a Predictor of Contribution Success. Our analysis reveals how project activity, measured primarily through commit frequency and maintainer responsiveness, predicts newcomer contribution success better than project size or age. This finding aligns with the work of Zhou and Mockus [45], who identified responsiveness as a factor in newcomer retention by demonstrating its critical importance specifically for time-constrained student contributors. A reason this was the case may be that projects with high activity levels often have better documentation, clearer processes, and more responsive communities, creating favorable conditions for successful student contributions. We recommend using commit frequency as a practical proxy metric to assess the likely responsiveness of an OSS community when guiding students in project selection. Our findings show that, for the students' onboarding, projects with more accessible and frequent collaboration are more important than project maturity.

Cosentino et al. [5], observed that project "healthiness" indicators should include not just size and popularity, but also activity patterns and community responsiveness. Our findings strongly support this broader conception of project health, particularly for assessing suitability for newcomer contributions. Projects demonstrating regular activity typically maintained better onboarding resources and more engaged communities, which proved particularly valuable for students navigating their first contributions.

Our findings also corroborate those of Fronchetti et al. [12], who found that project activity metrics, such as time to review pull requests, are important in attracting newcomers to OSS projects. Our observations extend this understanding by highlighting that responsiveness is particularly crucial not only for reviewing pull requests but also for helping contributors overcome barriers to contributing. In some cases, timely responses to questions about development environment setup or code architecture proved decisive in enabling students to make successful contributions despite technical complexity.

Students in our study were specifically instructed to avoid large, popular, and active projects to prevent disruptions, but at the same time to choose an engineered, collaborative, and active software project. This might have caused some students to pick projects that were not very active. For educational contexts, these implications are important. Prioritizing active projects with responsive maintainers may yield substantially higher success rates for student contributions than selecting projects based primarily on technical characteristics or domain relevance. This is also in line with the findings of previous work on the integration of OSS projects in software engineering education [24] regarding the selection of OSS projects.

Are "Good First Issues" Actually Good for Students? Our findings also raise important questions about the effectiveness of "good first issue" (GFI) labels for student contributors. While six of our 13 student groups explicitly cited the availability of such labeled issues as a factor in project selection, our findings suggest that these labels did not consistently lead to successful contributions. This aligns with the observations of Tan et al. [38] who found that issues labeled as newcomer-friendly often lack adequate context or contain tacit knowledge requirements that create hidden barriers for newcomers. Steinmacher et al. [35] observed that many GFI-labeled issues assume a level of project familiarity that newcomers do not possess. Our findings support this observation, as students frequently reported struggling with issues that were ostensibly designed for newcomers but required substantial contextual understanding. For instance, a group noted that despite selecting a GFI-labeled task in TEAMMATES, they still encountered significant problems in understanding how their contribution fits into the broader architecture.

This suggests that newcomer tasks should not only be technically accessible but also contain clear contextual information and be actively monitored by mentors. Our data support this more comprehensive view of newcomer task design, as students who received timely guidance on their tasks, regardless of GFI labeling, reported more positive experiences and achieved higher success rates. The mismatch between GFI labeling and actual newcomer experience may be due to the understanding of project maintainers, who may underestimate the knowledge required to complete seemingly simple tasks. For student contributors in particular, who may lack the project background knowledge to fill knowledge gaps independently, this can transform supposedly accessible tasks into significant challenges. These findings suggest that educational programs should approach GFI labels with caution and consider additional factors when guiding students toward suitable contribution opportunities. Instructors may need to perform additional vetting of GFI-labeled issues to ensure that they truly align with student capabilities and course timeframes, rather than relying solely on project-provided labels.

Student-Specific Barriers and Experiences. Our study identified two previously undocumented barriers affecting student contributors: "Conflicting Mentor Guidance" and "Communication Channel Ambiguity". While these barriers might potentially affect other contributor types, they are also impactful in educational contexts where students engage with OSS through structured academic programs rather than organic community participation.

"Conflicting Mentor Guidance" emerged when students received contradictory feedback from different project mentors, creating uncertainty about the correct approach to address problems. This barrier highlights an issue in distributed mentorship models common in OSS projects [8]. While experienced developers might reconcile conflicting feedback through their own judgment, students with limited domain knowledge often struggle to determine which guidance to follow, leading to delays or misaligned contributions. This issue may be especially problematic for students because of their academic context, where they typically work under strict deadlines and expect authoritative guidance similar to what they receive in coursework. Ford et al. [11] observed that newcomers with less confidence in their technical abilities (a common characteristic of students) are more likely to be negatively affected by ambiguous or conflicting guidance, which further supports why this barrier may impact student contributors.

"Communication Channel Ambiguity" occurred when students encountered unclear or incomplete information on how to access project communication platforms, effectively isolating them from community support. While previous research has identified communication challenges related to language or cultural differences [33], this barrier specifically relates to structural access issues that prevent newcomers from even initiating communication. For students with limited exposure to the variety of communication platforms used in professional software development (Discord, Slack, mailing lists, etc.), this ambiguity created significant impediments to seeking help. This barrier relates to what Canfora et al. [3] termed as 'YODA', where they try to identify and recommend mentors to newcomers joining the project to enable them to onboard in a hassle-free way. Students may be particularly affected by this barrier due to their limited professional experience in navigating the diverse communication ecosystems found in OSS communities.

These barriers may be particularly significant for students due to their unique position within OSS projects. Unlike voluntary contributors, who might have more flexible timelines or professional developers with broader industry experience, students operate within academic courses with firm deadlines and evaluation criteria. This creates a distinct contribution context that shapes how students experience and respond to barriers.

Intrinsic Motivation and Contribution Persistence. Our finding that groups selecting projects based on personal interest demonstrated higher success rates in having their pull requests merged highlights the role of intrinsic motivation in helping students persist through contribution challenges. This aligns with self-determination theory [27] and extends work by von Krogh et al [43] on motivation in OSS contribution by demonstrating its particular importance for students operating under academic constraints. Gerosa et al. [13] found that newcomers whose motivations align with project values show greater persistence when faced with barriers. Our data supports this, as students who selected projects based on interest or alignment with their personal goals demonstrated greater resilience when encountering obstacles. For example, one group contributing to JabRef persisted through multiple rounds of feedback and revisions, explicitly citing their interest in the project's purpose as motivation for continued effort. This observation is related to the findings of Trinkenreich et al. [40], who identified the alignment between contributor goals and project opportunities as a key factor in successful OSS onboarding. For students whose contributions are partially extrinsically motivated by course requirements and grades, the addition of intrinsic motivation appears to provide the necessary resilience to overcome barriers to contribution.

5.1 Implications

Our results hold implications for both students, maintainers, and educators who want to teach similar courses.

Student Contributors. For students who seek to contribute to OSS, it's important to highlight the following rules. Given the scope of this study, these rules are targeted at student contributors; however, some may also apply to junior contributors.

- First, students should prioritize projects with high commit frequencies and responsive maintainer communities when selecting contribution targets, as we believe those characteristics helped students avoid barriers, allowing them to receive help from maintainers and have their pull requests merged.
- Second, while programming language familiarity provides a good start, students should recognize that it alone does not eliminate barriers related to domain complexity and architectural understanding. Students may need to invest additional time to understand the project architecture, regardless of their language proficiency. The presence of complexity barriers despite language familiarity indicates that effective onboarding requires more than just matching contributors with projects in familiar languages.
- Third, encouraging students to select projects aligned with their personal interests appears to be very helpful, as groups who reported intrinsic motivation reported less impactful contribution barriers.
- Fourth, students should identify and connect proactively with specific project mentors rather than relying on general community support. In line with the study by Trainer et al [39], forming direct relationships with established contributors can significantly improve onboarding experiences. Students who established direct communication channels with specific maintainers reported fewer barriers overall in our analysis.

Combined, these four recommendations will help set up students who are looking to contribute to open-source, and should help increase the success rate of student contributors.

Educators. For teachers who teach similar courses, our findings emphasize the importance of project selection. As Singh [31] noted, the success of educational OSS programs depends heavily on the selection of projects that can accommodate student timelines and learning needs. Our findings reinforce this observation and suggest specific metrics (commit frequency, response rates) that instructors might monitor when selecting potential projects. Second, educational scaffolding should prepare students for the interconnected nature of contribution barriers rather than address technical, interpersonal, and process challenges in isolation. This aligns with the recommendations of Feliciano et al. [9], who advocated for a comprehensive preparation that covers both the technical and social aspects of OSS contributions. Our findings suggest that such holistic preparation is indeed necessary, given the co-occurrence of different barrier types in student experiences. Third, allowing students greater freedom to select projects to foster intrinsic motivation may yield better results than a strict assignment based solely on technical criteria. This approach is supported by work from Murphy-Hill et al [19], who found that developer autonomy positively correlates with contribution quality and persistence in software development contexts.

OSS Project Maintainers. For OSS maintainers interested in attracting student contributors, our findings highlight the particular importance of responsive communication and clear access to community platforms. The significant impact of low response rates on student contributions suggests that projects might benefit from dedicated "office hours" or accelerated review processes for student contributions, acknowledging the time constraints under which students operate. In addition, explicit documentation of communication channels and contribution processes can help mitigate student-specific barriers identified in our study. Turzo et al. [41] proposed that OSS projects should develop specific "newcomer journeys" that recognize the varied backgrounds and needs of different contributor types. Our findings support this approach, suggesting that student contributors in particular benefit from structured pathways that account for their academic constraints and learning objectives. Maintainers should also be aware of the potential impact of conflicting guidance on student contributors. As Steinmacher et al. [35] noted, mentorship coordination is particularly important for newcomers with limited domain knowledge. Our identification of "Conflicting Mentor Guidance" as a distinct barrier reinforces this recommendation, suggesting that projects interested in student contributors should establish mechanisms for coordinating mentor feedback.

6 STUDY DESIGN TRADE-OFFS AND THREATS TO VALIDITY

We reflect on the principal limitations and methodological trade-offs of our study, following Robillard et al.'s recommendations for transparent reporting in software engineering research [25]. We first discuss major design decisions, their implications, and then enumerate other threats not framed as trade-offs.

Data Source: Student Reflections vs. Direct Observation. *Decision Point:* We analyzed structured student reflection reports as our main data source, rather than direct observation or interviews. *Alternatives:* Real-time observation, mixed-methods, or periodic diaries. *Trade-off and Rationale:* Reports fit the pedagogical structure, minimize intrusion, and capture contemporaneous student perspectives. However, they may omit unrecognized or forgotten barriers. *Implications:* Some subtle or transient challenges may go unreported. We partially mitigate this by triangulating subjective reports with objective outcomes (e.g., pull request status, project metadata) and using a structured reflection template to encourage comprehensive reporting.

Sample Size and Context: Educational Assignment vs. Naturalistic OSS Participation *Decision Point:* Our study focused on 13 teams in a single master's level course at one university. *Alternatives:* Larger or multi-institutional studies, or sampling voluntary OSS participation. *Trade-off and Rationale:* A single-course, small-sample design enables in-depth, consistent analysis and direct linkage to assignment context, but limits generalizability beyond this setting. The controlled environment minimizes confounding factors but sacrifices coverage of the full range of OSS education or volunteer contexts. *Implications:* Findings are best interpreted as representative of structured course-based OSS assignments. To support transferability, we provide detailed context and project criteria. Broader studies are needed for wider generalization.

Analytical Framework: Prior Taxonomies vs. Open Coding *Decision Point:* Coding was grounded in the framework of Balali et al. [2], while permitting emergent codes. *Alternatives:* Fully inductive (open) coding or strictly deductive analysis. *Trade-off and Rationale:* Using a prior taxonomy ensures alignment with established literature and facilitates comparison, but risks overlooking student-specific nuances. Allowing new codes adds flexibility, but introduces some subjectivity. *Implications:* Our hybrid approach increases both rigor and sensitivity to context. Consensus coding and detailed notes mitigate interpretive bias.

In addition to these explicit trade-offs, several broader threats remain:

Social Desirability and Course Incentives. Because the reports were graded and visible to instructors, some responses may reflect what the students believed was expected or academically appropriate, which may have led to under-reporting of negative or personal barriers.

Language and Communication Constraints. All reports were written in English, which is not the native language of most students. This may have limited the expression or nuance of some reflections, especially on personal or interpersonal topics.

Framing and Confirmation Bias. Students were prompted to reflect on barriers and introduced to a specific theoretical framework, which may have primed them to emphasize certain experiences or categories while overlooking others.

Missing Non-Participation Data. Our analysis does not include students who dropped the course, disengaged, or failed to submit reports. Barriers leading to non-completion are thus underrepresented.

7 CONCLUSIONS

This study identified and analyzed the barriers faced by students contributing to OSS projects as part of a structured academic assignment. Through a systematic examination of 13 student group reports, we observed that, while students encounter familiar technical and process-related barriers, they also face unique challenges, including conflicting mentor guidance and ambiguity in communication channels, which are not fully captured by existing barrier frameworks. Our findings show that project activity (commit frequency) is a stronger predictor of successful contribution than project size or age, and that students motivated by personal interest experience greater persistence and success. Future research should explore a broader range of educational contexts and adopt longitudinal or mixed-methods designs to better capture the evolution and resolution of barriers over time.

DATA AVAILABILITY

The materials in this research, including the assignment description and project characteristics, are available as supplemental material https://doi.org/10.6084/m9.figshare.30229774.

REFERENCES

[1] Sogol Balali, Umayal Annamalai, Hema Susmita Padala, Bianca Trinkenreich, Marco A. Gerosa, Igor Steinmacher, and Anita Sarma. 2020. Recommending Tasks to Newcomers in OSS Projects: How Do Mentors Handle It? Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3412569.3412571

[2] 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. 27, 3–6 (2018). https://doi.org/10.1007/s10606-018-9310-8

[3] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 2012. Who is going to mentor newcomers in open source projects? Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2393596.2393647

[4] Oihane Cereceda and Danielle Quinn. 2020. A graduate student perspective on overcoming barriers to interacting with open-source software. *FACETS* 5 (05 2020), 289–303. https://doi.org/10.1139/facets-2019-0020

[5] Valerio Cosentino, Javier Canovas Izquierdo, and Jordi Cabot. 2017. A Systematic Mapping Study of Software Development With GitHub. *IEEE Access* PP (03 2017), 1–1. https://doi.org/10.1109/ACCESS.2017.2682323

[6] Roberto Almeida Bittencourt Debora M. C. Nascimento and Christina Chavez. 2015. Open source projects in software engineering education: a mapping study. *Computer Science Education* 25, 1 (2015), 67–114. https://doi.org/10.1080/08993408.2015.1033159

[7] Felipe Ebert, Fernando Castor, Nicole Novielli, and Alexander Serebrenik. 2021. An exploratory study on confusion in code reviews. *Empir. Softw. Eng.* 26, 1 (2021), 12. https://doi.org/10.1007/S10664-020-09909-5

[8] Fabian Fagerholm, Alejandro Guinea, Jürgen Münch, and Jay Borenstein. 2014. The Role of Mentoring and Project Characteristics for Onboarding in Open Source Software Projects. *International Symposium on Empirical Software Engineering and Measurement.* https://doi.org/10.1145/2652524.2652540

[9] Joseph Feliciano, Margaret-Anne Storey, and Alexey Zagalsky. 2016. Student experiences using GitHub in software engineering courses: a case study. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2889160.2889195

[10] Zixuan Feng, Igor Steinmacher, Marco Aurélio Gerosa, Tyler Menezes, Alexander Serebrenik, Reed Milewicz, and Anita Sarma. 2025. The Multifaceted Nature of Mentoring in OSS: Strategies, Qualities, and Ideal Outcomes. In *18th IEEE/ACM International Conference on Cooperative and Human Aspects of Software Engineering, CHASE@ICSE 2025, Ottawa, ON, Canada, April 27-28, 2025.* IEEE, 203–214. https://doi.org/10.1109/CHASE66643.2025.00031

[11] Denae Ford, Margaret-Anne Storey, Thomas Zimmermann, Christian Bird, Sonia Jaffe, Chandra Maddila, Jenna L. Butler, Brian Houck, and Nachiappan Nagappan. 2021. A Tale of Two Cities: Software Developers Working from Home during the COVID-19 Pandemic. *ACM Transactions on Software Engineering and Methodology* 31, 2 (Dec. 2021), 1–37. https://doi.org/10.1145/3487567

[12] Felipe Fronchetti, Igor Wiese, Gustavo Pinto, and Igor Steinmacher. 2019. What Attracts Newcomers to Onboard on OSS Projects? TL;DR: Popularity. In *Open Source Systems*, Francis Bordeleau, Alberto Sillitti, Paulo Meirelles, and Valentina Lenarduzzi (Eds.). Springer International Publishing, Cham, 91–103.

[13] Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. 2021. The Shifting Sands of Motivation: Revisiting What Drives Contributors in Open Source. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE).* IEEE, 1046–1058. https://doi.org/10.1109/icse43902.2021.00098

[14] GitHub, Inc. 2024. The State of Open Source: Octoverse 2024. https://github.blog/news-insights/octoverse/octoverse-2024/. Accessed: September 22, 2025.

[15] Christoph Hannebauer and Volker Gruhn. 2016. Motivation of Newcomers to FLOSS Projects. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2957792.2957793

[16] Christoph Hannebauer and Volker Gruhn. 2017. On the Relationship between Newcomer Motivations and Contribution Barriers in Open Source Projects. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3125433.3125446

[17] Hao He, Minghui Zhou, Qingye Wang, and Jingyue Li. 2023. Open Source Software Onboarding as a University Course: An Experience Report. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. https://doi.org/10.1109/ICSE-SEET58685.2023.00037

[18] Christopher Mendez, Hema Susmita Padala, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Nupoor Patil, Anita Sarma, and Margaret Burnett. 2018. Open source barriers to entry, revisited: a sociotechnical perspective. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3180155.3180241

[19] Emerson Murphy-Hill, Da Lee, Gail Murphy, and Joanna McGrenere. 2015. How Do Users Discover New Tools in Software Development and Beyond? *Computer Supported Cooperative Work (CSCW)* 24 (07 2015). https://doi.org/10.1007/s10606-015-9230-9

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

[21] Debora M. C. Nascimento, Christina F. G. Chavez, and Roberto A. Bittencourt. 2018. The Adoption of Open Source Projects in Engineering Education: A Real Software Development Experience. IEEE Press. https://doi.org/10.1109/FIE.2018.8658908

[22] Fabio Palomba, Damian Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman, and Alexander Serebrenik. 2021. Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells? *IEEE Transactions on Software Engineering* 47, 1 (2021), 108–129. https://doi.org/10.1109/TSE.2018.2883603

[23] Gustavo Pinto, Clarice Ferreira, Cleice Souza, Igor Steinmacher, and Paulo Meirelles. 2019. Training Software Engineers Using Open-Source Software: The Students' Perspective. https://doi.org/10.1109/ICSE-SEET.2019.00024

[24] Gustavo Henrique Lima Pinto, Fernando Figueira Filho, Igor Steinmacher, and Marco Aurelio Gerosa. 2017. Training Software Engineers Using Open-Source Software: The Professors' Perspective. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. https://doi.org/10.1109/CSEET.2017.27

[25] Martin P Robillard, Deeksha M Arya, Neil A Ernst, Jin LC Guo, Maxime Lamothe, Mathieu Nassif, Nicole Novielli, Alexander Serebrenik, Igor Steinmacher, and Klaas-Jan Stol. 2024. Communicating study design trade-offs in software engineering. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–10.

[26] Paige Rodeghero. 2020. An Exploratory Field Study of Programmer Assistance-Seeking during Software Development. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 169–172. https://doi.org/10.1145/3387940.3392237

[27] Richard Ryan and Edward Deci. 2000. Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *The American psychologist* 55 (01 2000), 68–78. https://doi.org/10.1037/0003-066X.55.1.68

[28] Italo Santos, Katia Romero Felizardo, Igor Steinmacher, and Marco A. Gerosa. 2025. Software solutions for newcomers' onboarding in software projects: A systematic literature review. *Information and Software Technology* 177 (2025), 107568. https://doi.org/10.1016/j.infsof.2024.107568

[29] Italo Santos, Katia Romero Felizardo, Bianca Trinkenreich, Daniel M. German, Igor Steinmacher, and Marco A. Gerosa. 2025. Exploring the Untapped: Student Perceptions and Participation in OSS. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering* (Clarion Hotel Trondheim, Trondheim, Norway) *(FSE Companion '25)*. Association for Computing Machinery, New York, NY, USA, 859–870. https://doi.org/10.1145/3696630.3727243

[30] Jefferson Silva, Igor Wiese, Daniel M. German, Christoph Treude, Marco Aurélio Gerosa, and Igor Steinmacher. 2020. A theory of the engagement in open source projects via summer of code programs. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3368089.3409724

[31] Vandana Singh. 2012. Newcomer integration and learning in technical support communities for open source software. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2389176.2389186

[32] Therese Mary Smith, Robert McCartney, Swapna S. Gokhale, and Lisa C. Kaczmarczyk. 2014. Selecting open source software projects to teach software engineering. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2538862.2538932

[33] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2675133.2675215

[34] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming open source project entry barriers with a portal for newcomers. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2884781.2884806

[35] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco A. Gerosa. 2018. Almost there: a study on quasi-contributors in open source software projects. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3180155.3180208

[36] Igor Steinmacher, Igor Scaliante Wiese, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2014. The hard life of open source software project newcomers. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2593702.2593704

[37] Xin Tan, Yiran Chen, Haohua Wu, Minghui Zhou, and Li Zhang. 2023. Is It Enough to Recommend Tasks to Newcomers? Understanding Mentoring on Good First Issues. arXiv:2302.05058 [cs.SE] https://arxiv.org/abs/2302.05058

[38] Xin Tan, Minghui Zhou, and Zeyu Sun. 2020. A first look at good first issues on GitHub. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3368089.3409746

[39] Erik H. Trainer, Chalalai Chaihirunkarn, Arun Kalyanasundaram, and James D. Herbsleb. 2015. From Personal Tool to Community Resource: What's the Extra Work and Who Will Do It? Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2675133.2675172

[40] Bianca Trinkenreich, Igor Wiese, Anita Sarma, Marco Gerosa, and Igor Steinmacher. 2022. Women's Participation in Open Source Software: A Survey of the Literature. *ACM Transactions on Software Engineering and Methodology* 31, 4 (Aug. 2022), 1–37. https://doi.org/10.1145/3510460

[41] Asif Kamal Turzo, Sayma Sultana, and Amiangshu Bosu. 2025. From First Patch to Long-Term Contributor: Evaluating Onboarding Recommendations for OSS Newcomers. arXiv:2407.04159 [cs.SE] https://arxiv.org/abs/2407.04159

[42] Elaine Venson and Reem Alfayez. 2024. Bridging Theory to Practice in Software Testing Teaching through Team-based Learning (TBL) and Open Source Software (OSS) Contribution. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '24)*. ACM, 72–81. https://doi.org/10.1145/3639474.3640081

[43] Georg Von Krogh, Stefan Haefliger, Sebastian Spaeth, and Martin W. Wallin. 2012. Carrots and rainbows: motivation and social practice in open source software development. 36, 2 (2012).

[44] Minghui Zhou, Qingying Chen, Audris Mockus, and Fengguang Wu. 2017. On the scalability of Linux kernel maintainers' work. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3106237.3106287

[45] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: willingness and opportunity in OSS community.