How Open Source Software Used in Software Development Education Affects Students, Professors, and the Communities They Join

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Johnathan Eftink

Spring 2025

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Kent Wayland, Department of Engineering and Society

Introduction

Free and Open Source Software (FOSS), while it may be used widely in the corporate world, is underutilized in education. In this context, FOSS in education refers to students collaborating and adding to an open-source project, not using an open-source Learning Management System (LMS) such as “Moodle” (Lakhan, Jhunjhunwala 2008). When the students collaborate on these FOSS projects, they are participating in group discussion as well as reading, adding, and suggesting changes in code for these projects. Some researchers such as Pinto et al. (2019) have used FOSS in classroom settings around the world to show the positive effects of FOSS in the classroom. From this research we can see the importance of FOSS in education, but the social system that this takes place in, and its effects are yet to be fully understood. Understanding how FOSS used in software development education affects students, professors, and the communities they join can lead to a more effective and collaborative learning environment, as all those involved will be able to better set their expectations. This is why it is important to understand *how FOSS used in software development education affects students, professors, and the communities that they join*.

Currently, not much is known about how students, professors, and communities interact because current research only focuses on student exit interviews. This means that there is a gap in the research in which the social system is described for students. By filling in this research gap and using social theories to understand this social system, professors will be able to help students understand how they should interact with the contributors. Proper interactions will then allow students to be better integrated into these FOSS groups and therefore do better in their coursework and have even more increased self-esteem than Salerno et al. (2023) describe in their

paper. All of this shows that there is a gap in understanding socially how students, professors, and communities interact, but by filling in this gap, students will be able to perform better and experience greater outcomes.

Background

Free and Open Source Software (FOSS), while it may be used widely in the corporate world, is underutilized in education. To be a piece of "free and open source software", means that it is open for the public to use, distribute, and even change freely (RedHat). FOSS, as described by Yang and Wang (2008), is software that has many contributors to a project and contributions can be done by anyone. This open-source movement gained momentum in the 1980s with the GNU Project (FOSSID, n.d.). The GNU project's goal was to completely build an operating system using only open-source parts and was eventually able to do so. This goal of an open-source operating system was continued by the Linux Foundation in the 2000s with their UNIX based operating systems. The Linux Foundation would help grow projects both based on operating systems and those that were not. This helped create a larger culture of FOSS and led to many developers participating in FOSS in their free time (FOSSID, n.d.).

While anyone can make contributions to FOSS projects, there is still an approval process in which a core community will approve of your changes. Adam Alami et al. (2020) discuss this in which they describe the process of making a change through a "pull request." It is up to the community to decide whether to accept this request and therefore make a change to the project. This community that would accept requests tends to have a very small group of core contributors, as described by Jeff et al. (n.d.). It is important to note, however, that not all projects have a very small core group. For instance, an open-source Operating System (OS) project, Fedora, has estimated that they have 2000 contributors, which in terms of development

teams is quite large (Flock Kraków, 2016). This shows that while many projects do have a small set of contributors, there are larger projects which have a large set of contributors. So, while education does often use FOSS, rarely are students able to make changes to a FOSS project within an educational setting. For the courses that do allow students to participate in FOSS projects, they are either assigned or will choose to join a specific FOSS project. These FOSS projects will vary in size, some might involve them contributing along with hundreds, or some teams might be within single digits of size (Stone, 2016). Then on, they will learn from/contribute to said project during their educational course. Their professors will then evaluate them, often based on their contributions and how they align with the course's learning objectives.

FOSS projects often have a set of developers that are core to them and focus on that project, but then also a set of developers that are temporary or not fully committed (e.g. VSCodium has a main developer that will push changes but also other developers that will make smaller changes from time to time). For FOSS in software development education, the students and professors are part of the group that is temporary to the project. The students and professors can become part of the "core" group, but purely within the context of a course, they are only temporary. This core group will normally have a set of standards that they expect other contributors to follow such as standards for communication. For instance, messages written by and for contributors might have to be of a certain quality, through their desired medium (e.g. GitHub or Discord), and follow patterns they deem fit. Specifically, a FOSS group's issue reports might be expected to have a certain tact to them and focus on reporting problems within the code, rather than requesting help with installation issues (bobitetly). These standards set by the contributors can be made through group discussion and often vary from group to group.

Just like any other person that joins a FOSS project, students and professors must learn and follow the standards of the individual projects that they are on. If a project does not allow you to change the code until you create some sort of bug report, then you must do so (bobitetly). However, it would be completely reasonable for a different project to require a request to change code before you can bring a report to their attention. Having to learn these standards and having such a variety of standards allows students to gain professional experience, which is what the FOSS based software development courses wish to do. Through all of this, FOSS projects can create a community of differently skilled, but goal aligned individuals that students and professors are able to join. Education-based individuals can then also change the culture around these FOSS projects, thus evolving the culture and standards of professionals.

Literature Review

Based on the FOSS project, there will be different types of cultures and policies that must be learned by those joining the project. An important culture to understand is the one that develops between the user and contributor. This is described by Matsumoto et al. (2008) in which they concluded that a key factor in the success of FOSS projects is the ability for developers and users to collaborate closely with each other. When there is a disconnect between the user and developer, this will drive usability down as the users' needs will not be met as much as they may desire. When cultural or technical differences start to arise between contributors and users, this can lead to the project splintering and turning into multiple distinct groups instead of a cohesive effort by all contributors. For instance, if one developer believes that a certain new feature should be added for a certain user base, but the original developer believes that this new feature would distract from the true goal, the first developer might splinter off and create their

own group. So, it is in the best interest of FOSS groups to align in that group's culture to produce the best product for users and to stay unified throughout the process.

Recent research has begun to highlight the psychological and practical impacts of FOSS on student participants such as Salerno et al. (2023). Salerno et al. investigated how students' view of themselves changes when they are the contributors. This research shows that students experience positive psychological effects while working on open-source projects and therefore show FOSS's importance in education. To show some of the practical effects developed through FOSS Pinto et al. (2019) examined students as contributors, exploring how these students learned to communicate and developed skills with different technologies such as Linux within the FOSS ecosystem that they participated in. This shows that students were able to develop practical skills through their participation in a FOSS system. Overall, these sources show the usefulness and importance of FOSS within software development education.

It is also important to understand the role of the professor within this system. Research by Silva et al. and Pinto et al. (2019, 2017) highlights the difficulties professors face with FOSS-based homeworks, its impact on their classes, and the potential drawbacks of using FOSS. These are important as it explains how professors work and react within a FOSS system as well as how actualizable a FOSS based class or homework is. It is also the duty of the professors to investigate how to involve students in FOSS projects. Professors also must help students choose projects in which they are wanted as emphasized by Ellis et al. (2011). A FOSS project requires collaboration with the contributors of the project, so both the student and the FOSS group must want the student to join. This shows the importance of making sure your cultures match with whomever your group decides to join.

Methods

For my research, I am collecting journal articles about software development education using FOSS, and information about FOSS groups and cultures. Articles about software development education that use FOSS are specifically about how students/professors feel, their experiences, any difficulties, and any outcomes that might be observed from using FOSS. Articles about FOSS groups and cultures are about how people subscribe to/join a FOSS community, what FOSS cultures look like, and general experiences of the groups that are involved in FOSS. It is important to note that the FOSS research will not be heavily involved in the technical aspect of FOSS. For instance, research about how the FOSS development process works and tools they might use will not be covered. Overall, these articles are in two main classes: FOSS in education and FOSS's social implications on people.

Most of these articles were obtained either from Google Scholar or Web of Science/Web of Knowledge through keyword searches such as "FOSS communities," "FOSS cultures," "FOSS in education," "software development education with FOSS." To ensure these articles are from reliable sources, they were gathered from published journals: ACM and IEEE. The ACM is the Association for Computing Machinery which is "the world's largest computing society" and one of their goals is to help promote research (Association for Computing Machinery, n.d.). ACM publications are all peer reviewed, meaning that the research from them has been rigorously vetted. Many articles used throughout are also from IEEE (Institute of Electrical and Electronics Engineers) which, like ACM articles, are all peer reviewed.

Results

In trying to understand how students joining FOSS can affect the communities they look to contribute to, we must first understand communities of practice. Canada's National Library of Medicine describes it as a group of people that share a common interest and come together to fulfill both individual and group goals (Richard et. al.). The community we are focusing on is one in which students will be entering within an academic setting, but the contributors to this project are not participating because of academia. This means there will be an inherent difference between the student and the professional. These students must experience joining a skilled community and the skilled community will have to learn how to accept people with less experience than them.

To get a good understanding of the community of practice that students are joining, it is easy to compare it to a community of a regular workplace. It is important to note these professional FOSS contributors are not expected to make contributions like people who are employed on a project are expected to make contributions. FOSS contributors are able to make contributions when they can and leave as they please, whereas in the workforce there is less freedom. While this may lead to an altered community of practice from that of a regular workplace, the community of practices is largely the same. The communities still are a collection of individuals that share a common interest, are skilled within their interest, and have both individual and group goals.

To understand how a student might feel when joining a community of practice, we can look at an employee joining a new company and draw comparisons between the two situations. An example of workers joining a new community of practice is in research by Janet Holmes (2015) in which a number of New Zealand workers join new jobs. Some experiences described by these workers were that many had to develop new behaviors or change their life space in

order to cope with the change in their professional life. For instance, a New Zealander describes immigrating to the UK and using profanity, which while he was used to it, the UK staff was not and were taken aback. While this is a more extreme example, it shows that communication is key to joining a community of practice and can lead to some frustrations when it is not understood by all.

Communication is also key in FOSS as shown by Salerno et al. (2023) and their research into students' view of software education through FOSS describes the documentation problem by discussing student's issues with documentation. They describe that at the beginning of the course, only 16% of students were expecting to run into issues with clarity and understanding of documentation, but they found that 27% of students had run into issues. Having more than one in four students run into issues with communication shows its importance within a community of practice and how crucial it is for an incoming student to learn how the group communicates and how to speak the group's "language." Morgan and Jensen (2014) described the frustrations that students can have during this process and how the students find the process of communication to be taxing. All of this shows that joining the community of practice revolves around communication and many frustrations can arise from it.

However, to much of the group, the joining phase for the student is not the beginning of their community, but well into its inception. This means that a different "phase" of the community of practice must be studied to understand how individuals feel. Borzillo et al. (2011) describe a community of practice forming within different phases, of which, the allocation and later architectural phase is where individuals that are outside of the "core" members, become part of the core. For a FOSS project, this would mean that people join the community and look to contribute (which is exactly what the students do as part of their courses). Borzillo describes

either a pull (where core members invite people into the group) or a push (where people ask a core member if they can join). FOSS projects will mainly be of this second type, especially for students. Students that join will ask to become part of the core group, not by directly asking, but instead by beginning to participate in discourse and by requesting to make changes to code. Of course, for different groups this joining process might be done differently, but for most FOSS groups, this is how students will be able to join.

Students will try to become core members during a time called the architectural phase. In this phase, people outside of the core will create subtopics that will allow them to become part of the core (Borzillo et al., 2011). In FOSS groups, this subtopic would be a new functionality or even a new set of bug fixes. These subtopics help legitimize the student in the group. While before the student might just be taking interest in a project, by starting to create their own requests for changes, they are legitimizing their desire to be part of the core. This allows the core to either reject the students' changes and expertise if they believe it is not following the rules and ideals of the community of practice. If the student is allowed to make changes, then the community has decided that they follow the rules and ideals enough to become part of the community. It is important to note that for some projects, it might be a single leader that decides these changes. A singular leader will decide if the changes are deemed "good enough" to be part of the project and the contributor to become part of the core.

Throughout this process, the core group of people is deciding whether or not to let the student become part of the core. Therefore, it is only on the core's terms that someone is able to join. This means that the core is going to view the person as a positive addition to the group since, if they are received negatively, they would not allow them to join. This shows that the process that allows a student to be will integrated into a FOSS group and become a core member

of the group is to: 1) become an outlying member of the group through reading documentation to understand the group's goal, 2) read through the code that is published to understand the ideals the group might have (such as naming conventions and overall structure), 3) produce one's own changes they want added to the FOSS project, making sure it aligns with the group goal, and allowing the student to become part of the core, 4) to stay within the core, the student must stay in communication with core members, must continue to contribute to code in some way (either through participating in discourse or continuing to add their own code). With these steps, a student can seamlessly join a group, leading them to become a member of the community of practice and therefore allowing them to improve the FOSS group as a whole and not leading to frustration through lack of communication or knowing the group's goals.

During this process, it is crucial for professors to adequately prepare their students for joining FOSS communities. While professors grade students and influence their contributions, their role is primarily supportive and indirect. Professors guide students in navigating these communities, but they rarely engage directly with the FOSS projects themselves. This makes the professor's influence more about how students communicate and contribute, rather than shaping the FOSS community itself. To the FOSS contributors, professors are just an extension of the students' involvement. However, for the students, professors represent an additional layer of community and guidance that coexists with the FOSS community. Therefore, it is on the professor to allow their academic community to elevate the students in their existence within the FOSS community. Professors, by educating their students on the four steps to become a core member and what ideals a FOSS community might have, are able to alleviate frustrations that students might experience with their FOSS community and allow FOSS communities to easily accept these students.

Conclusion

In conclusion, Free and Open Source Software (FOSS) offers significant benefits and challenges within the context of software development education. For students, engaging with FOSS projects provides valuable hands-on experience and a deeper understanding of real-world software development practices. Through these experiences, students not only develop technical skills, but also learn how to navigate the social dynamics of open-source-software communities. The process does come with its difficulties, however, as students must adapt to varying communication styles and norms which can lead to frustration from all involved.

For professors, incorporating FOSS into the curriculum presents both opportunities and challenges. Professors play a critical role in guiding students through these projects by helping them navigate the complexities of collaboration within communities of practice. The integration of FOSS into coursework requires careful planning to ensure students can both contribute meaningfully and meet course objectives. Despite these challenges, FOSS can enhance the learning experience by offering students a chance to engage in real-world software development processes.

Finally, the communities that students join through FOSS are shaped by their contributors. As students bring fresh perspectives to these projects, they also face the challenge of adjusting to established norms and practices within the community. This interaction not only benefits the students but also helps drive innovation within the FOSS ecosystem. The process of becoming part of a FOSS community teaches students valuable lessons about communication, collaboration, and professional growth, all of which are valuable in the professional world.

Overall, FOSS in software development education bridges the gap between academic learning and professional practice. While the process can be challenging for both students and professors, the opportunity to contribute to and learn from open-source projects makes it a valuable experience. By understanding the social dynamics and communication barriers that come with participating in FOSS, students, professors, and community members can work together more effectively to create a more collaborative and innovative learning environment.

Works Cited

Alami, A., Cohn, M. L., & Wąisowski, A. (2020). How do foss communities decide to accept pull requests? *Proceedings of the Evaluation and Assessment in Software Engineering*, 220–229. https://doi.org/10.1145/3383219.3383242

Association for Computing Machinery. (n.d.). *About the ACM organization*. About the ACM Organization. https://www.acm.org/about-acm/about-the-acm-organization

Azarbakht, A., & Jensen, C. (2013). Drawing the big picture: analyzing FLOSS collaboration with temporal social network analysis. *Proceedings of the Doctoral Consortium at the 9th International Conference on Open Source Systems, Koper-Capodistria, Slovenia, 2013*, 9–21.

bobitently, (Feb, 2018). *Only allow issues to be created by contributors or people who have created pull requests #293*, deag-github. GitHub. https://github.com/dear-github/dear-github/issues/293

Borzillo, S., Aznar, S., & Schmitt, A. (2011). A journey through communities of practice: How and why members move from the periphery to the core. *European Management Journal*, *29*(1), 25-42. https://doi.org/10.1016/j.emj.2010.08.004

Ellis, H. J., Hislop, G. W., Chua, M., & Dziallas, S. (2011). How to involve students in Foss Projects. *2011 Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/fie.2011.6142994

FOSSID. (n.d.). *Open source software explained - history, Benefits & Perils*. FossID.

https://fossid.com/articles/unlock-the-full-potential-of-open-source-with-fossid/

Jeff, B., Preston, J., Qu, J., & Hayes, J. An Examination of Collaborative Development Patterns of FOSS Repositories.(n.d.)

Linde, A., & Linderoth, H. C. (2006). An actor network theory perspective on IT projects. *Making Projects Critical*, 155–170. https://doi.org/10.1007/978-0-230-20929-9_8

Matsumoto, S., Kamei, Y., Ohira, M., & Matsumoto, K. I. (2008). A comparison study on the coordination between developers and users in FOSS communities. *Socio-Technical Congruence (STC 2008)*, (8), 1-9.

Miller, M. (n.d.). State of fedora, flock 2018 (PDF) - Matthew Miller. https://mattdm.org/fedora/2018flock/2018-State-of-Fedora.pdf

Morgan, B., Jensen, C. (2014). Lessons Learned from Teaching Open Source Software Development. Corral, L, Sillitti, A, Vlasenko, J, Wasserman, Al. *Proceedings of the 10th IFIP WG 2.13 International Conference on Open Source Systems (OSS). (133-142) Springer-Verlag Berlin https://link.springer.com/chapter/10.1007/978-3-642-55128-4_18*

Opensource.com. (n.d.). Red Hat Inc. *What is open source?* https://opensource.com/resources/what-open-source

Pinto, G. H., Filho, F. F., Steinmacher, I., & Gerosa, M. A. (2017). Training software engineers using open-source software: The Professors' Perspective. *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. https://doi.org/10.1109/cseet.2017.27

Pinto, G., Ferreira, C., Souza, C., Steinmacher, I., & Meirelles, P. (2019). Training software

engineers using open-source software: The students' perspective. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. https://doi.org/10.1109/icse-seet.2019.00024

Richard, L., Chiocchio, F., Essiembre, H., Tremblay, M. C., Lamy, G., Champagne, F., & Beaudet, N. (2014). Communities of practice as a professional and organizational development strategy in local public health organizations in Quebec, Canada: An evaluation model. *Healthcare Policy, 9*(3), 26-39.

Salerno, L., de França Tonhão, S., Steinmacher, I., & Treude, C. (2023). Barriers and self-efficacy: A large-scale study on the impact of OSS courses on student perceptions. *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, 320–326. https://doi.org/10.1145/3587102.3588789

Silva, F. G., Brito, M. S., Tavares, J. V., & Chavez, C. von. (2019). Floss in software engineering education. *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, *18*, 234–243. https://doi.org/10.1145/3350768.3353815

Sstone, Z. (2016). *Celebrating Tensorflow's first year.* Google Research. https://research.google/blog/celebrating-tensorflows-first-year/

Yang, J., & Wang, J. (2008). Review on free and Open source software. *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, *1*, 1044–1049. https://doi.org/10.1109/soli.2008.4686552

Warf, B. (2015). Networks, geography of. *International Encyclopedia of the Social & Behavioral Sciences*, 567–571. https://doi.org/10.1016/b978-0-08-097086-8.72123-4

Wen, S.-F. (2018a). Learning secure programming in Open source software communities: a socio-technical view. *Proceedings of the 6th International Conference on Information and Education Technology*, 25–32. https://doi.org/10.1145/3178158.3178202