Student Software Engineering Learning in HFOSS Projects

Dr. Becka Morgan, Western Oregon University

Becka Morgan takes great joy in teaching students computing languages, a subject she has been passionate about since she learned to program in 2006 as a non-traditional student. She is driven to create an inclusive environment. Consequently Dr. Morgan was drawn to teaching FOSS and HFOSS development based on work that is being done that suggests underrepresented groups are attracted to HFOSS participation. She teaches a one-term HFOSS course to both senior and graduate level students. The goal of the course is to engage all students in participation that ranges from improving documentation to submitting patches. Learning to teach students how to participate in HFOSS is an ongoing process. As part of the continuing efforts to pursue that knowledge Becka is a graduate of the 2013 and 2016 POSSE workshop and has trained to be a facilitator.

Heidi J.C. Ellis, Western New England University

Heidi Ellis is a Professor in the Computer Science and Information Technology department at Western New England University. Dr. Ellis has a long-time interest in software engineering education and has been interested in student participation in Humanitarian Free and Open Source Software (HFOSS) since 2006.

Dr. Gregory W Hislop, Drexel University

Gregory Hislop is a Professor and Senior Associate Dean in the College of Computing and Informatics at Drexel University. His scholarly interests span computing education research, information technology for teaching and learning, and software engineering. Prior to coming to Drexel, Dr. Hislop spent 18 years working in government and industry, where his efforts included software development and support, technology planning and evaluation, and development and delivery of technical education.

Student Software Engineering Learning in HFOSS Projects

ABSTRACT

Humanitarian Free and Open Source Software (HFOSS) projects are supported by communities that focus on developing software to solve societal challenges and improve the human condition. These projects provide rich opportunities for computing students to practice and learn both technical and professional skills. In addition, the transparency of HFOSS projects provides students with an opportunity to create a portfolio of their contributions to real-world projects. This paper reports on three different undergraduate courses where students learned by participation in an HFOSS project. The paper provides an overview of each class and description of results. Student reflective writing about their class experiences was used to gather unstructured observations about the student experience and learning. This student perspective is summarized and discussed to provide insight into the effect of student participation in HFOSS projects as part of an undergraduate computing program.

KEYWORDS

HFOSS, Humanitarian Computing, Free and Open Source Software, Computing Education

1 INTRODUCTION

Free and Open Source Software (FOSS) is software that is developed transparently with source code and other artifacts accessible and a license such that anyone can study, modify, and share the software. FOSS has gained a significant market share across a range of applications including operating systems, cloud computing, databases, and big data. In fact, the majority of enterprise, mid-market, and small businesses are widely adopting FOSS [1]. In addition, many major companies, including proprietary software vendors such as IBM and Microsoft, have groups devoted to open source.

Humanitarian FOSS (HFOSS) is FOSS that exists to improve the human condition. HFOSS projects address domains including disaster response, micro-finance, education, and health care. The altruistic objectives of HFOSS projects are intriguing for computing education since some studies have shown that the software applications that help people may attract female students to computing. [2, 7, 8, 12, 14, 21, 23]. In addition, the humanitarian nature of HFOSS communities appears to create a welcoming environment for new contributors.

FOSS projects have several characteristics that make them particularly attractive and important to integrate into student learning about software engineering:

- Transparency - All source code, and many software project processes, communications, documentation, and other artifacts are accessible, allowing students to view the project at different stages of maturity while also providing the opportunity for their own contributions to be made visible.
- Openness - The open licenses used in FOSS projects avoid the issues of proprietary development and intellectual property that may occur when students participate in proprietary projects.

- Scale and complexity - Successful FOSS projects are complete software products. They have a large code base, defects and feature requests, defined processes and tool chains, developer teams (often distributed), documentation, clients who use the product, etc. Involvement in an ongoing FOSS project exposes students to the complexity and scale of a real-world project.
- Innovation - FOSS has become the source of many innovations in software engineering practice. This makes it more important for students to gain an understanding of FOSS tools, techniques, and culture.

These characteristics are a natural fit for instructors who want to utilize real-world projects as a basis for student learning about software engineering [3, 6, 13, 16, 22, 27, 30].

The characteristics of open source projects create potential for student learning across a wide range of topics including:

- Technical Skills - The opportunities begin with programming, debugging, testing in the context of a large code base, and extend to include development tools, version control, product packaging, etc.
- Professional Skills - By observing and interacting with an HFOSS community, students can develop skills in team participation, critical thinking, and communication. Interactions with an HFOSS project provide students with the opportunity to observe both good and bad examples of these skills.
- Distributed Development - Many FOSS projects have contributors from around the world. Student participation can provide an opportunity to work with a globally distributed software team.
- Software Process - The software development process used by the HFOSS community is usually agile and HFOSS projects are often early adopters of new development approaches such as continuous integration, and containerization. Indeed, many applications that support these approaches such as Ansible, and Docker are open source projects themselves.
- Computing for Social Good - Participation in an HFOSS project provides students with some understanding of the potential for positive social impact of computing.

While the potential for student learning from HFOSS is great, there are challenges to taking this approach. There are typically multiple learning curves for both student and instructor including tools, development approaches, and project application knowledge. HFOSS projects are a less-structured environment than a typical classroom and some students have difficulty adjusting to a less-ordered environment. Instructors face additional challenges related to the creation and grading of assignments within the ever-changing environment of an HFOSS project. Instructors also encounter the difficulty of identifying appropriate projects and appropriate tasks within a project [32]. FOSS project selection for student involvement is an ongoing area of research [17, 20]

2 BACKGROUND

This section presents a brief summary of related work and the motivating questions for

this paper.

2.1 Related Work

2.1.1 *Pedagogy.* Many open source projects are built around communities, and the structure has often been described in terms of the concepts of Communities of Practice. Wenger defines communities of practice (CoP) as "groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly" [35]. This is a useful framework for FOSS, which operates under the philosophy that people working in groups, sharing code and mutually looking for and fixing bugs, creates an environment where the improvement of code takes place dramatically faster and more efficiently then solitary work [33].

Within this framework of Communities of Practice, student participation can be viewed as fitting the notion of legitimate peripheral participation. This allows for new participants to join the project, initially contributing in small ways, and gradually work from the periphery to more active core participation. Student participation can also be viewed as a form of cognitive apprenticeship [9] and fits easily with notions of active learning, and problem-based learning [19]. For HFOSS, the community can help provide a support system of experts with a variety of backgrounds [28]. In addition, [34] concludes that such collaboration can help reduce the impact of gender stereotype.

2.1.2 *HFOSS in Education.* Open source software has been used as a basis for student software engineering learning since the late 1990's [31]. A common approach is to utilize a FOSS project as the basis for a capstone project [3, 10]. One obvious way for students to participate in a FOSS project is via code contributions. However, there are a wide variety of ways that students can contribute ranging from documentation to testing to design [25]. Student participation in HFOSS can be closely aligned with service learning [26], an aspect of computing that has been shown to be attractive to groups under-represented in computing [4, 24].

The HFOSS Project was started in 2006. The HFOSS project was started in 2006 as a collaborative effort across several institutions to determine if "doing good" via participation in open source software projects could draw more students to computing disciplines [29]. Initial investigations into student learning in HFOSS identified:

- Potential roadblocks to student participation [5]
- The need for support for instructors [11]
- Guidelines to aid instructors [15]
- An initial approach to project selection [17]

Other research has explored ways to support student involvement in HFOSS projects. Results of this research [13, 14, 16, 18] have shown that student involvement in HFOSS has positive impact on:

- Student attitude towards software engineering

- Self-reported learning of software engineering knowledge
- Selection of major and career plans
- Technical and professional knowledge
- Understanding of how to develop and maintain software

It should be noted that female students reported a significantly greater understanding of how to manage software projects [14].

2.2 Motivating Questions

The investigation reported in this paper seeks to extend and confirm the results related to student learning reported above. The method is to examine three undergraduate courses that each had a focus on student participation in an HFOSS project. The courses were taught at three institutions by three different instructors. The instructors all have significant experience with involving students in HFOSS projects. The focus of this investigation can be summarized as an attempt to deepen understanding of student participation in HFOSS communities with regard to the following questions:

1) What types of knowledge and skill do students report developing by working on an HFOSS project?
2) Do students indicate that HFOSS participation provides motivation or affects confidence about pursuing computing careers?
3) What impact does it have on students to interact with the development community of an HFOSS project?

A key difference in the method of this paper is that it focuses on reflective writing of the students. Earlier work relied on structured surveys with Likert scale items and some opportunity for student comment on selected topics. The approach for this investigation was to collect student thinking without the prompting on particular aspects of expected learning and see what topics emerged naturally from the students' reflections.

In the sections that follow, each of the courses is summarized including an overview of the course, description of the student project, summary of results of student participation, and student observations on their learning. These sections are followed by a discussion of the themes that emerge from the unstructured student reflections.

3 COURSE EXPERIENCES

This section describes the three courses that were taught with an emphasis on student learning by participation in HFOSS projects and student reflective writing on their experiences.

3.1 Western Oregon University

Western Oregon University is a public, mid-sized university with a total enrollment of 5,382 (4,833 undergraduate and 549 graduate students) located in Monmouth, Oregon. The Division of Computer Science houses both the Department of Computer Science and the Department of Information Systems. The Department of Computer Science has approximately 145 students

and the Department of Information Systems has approximately 60 students.

3.1.1 *Course Overview.* CS435 Open Source Software Development is a senior level course offered during Spring, a 10 week term, every year as an elective choice for the Computer Science major. Approximately 90% of WOU seniors take Open Source Software Development each year.
The course starts with specific assignments designed to provide an overview of what open source software is, cover the tools of the community, and set up communication tools to record the journey of each student. Students self-select into groups of 3 - 4 that will serve as their team during the course. After this introduction, students learn about evaluating open source projects, triaging bugs, and how to use bug-tracking systems. All of these beginning assignments are used to learn about the FOSS community as a whole and were developed and shared through the site foss2serve.org. The rest of the term is used to actively participate in finding, fixing, and submitting patches for bugs within the Mozilla community.

3.1.2 *Project.* CS435 students participate with the Mozilla project, working on the Developer Tools (dev tools) or the Mozilla-central core. Mozilla was chosen as the project for this class based on its reputation as being an open, friendly community and for its work with student populations. Members of the Mozilla community have been guest speakers in several offerings of the course. In addition, a member of the Red Hat community comes to speak about licensing each year.

Mozilla also provides a variety of different areas in which to participate. This allows students to start in dev tools, a smaller area with a more manageable environment to build as a newcomer than the mozilla-central core. The work with the dev tools environment gives many students the confidence to move onto the central core.

3.1.3 *Results.* In the three years Mozilla has been used as the open source project for the CS435 class, fifteen out of sixteen groups have submitted a pull request for between one and three bugs. During the 2018 spring term all five groups had their pull requests merged into the code base. Additionally, individuals have also attempted to patch a bug solo after the group bug was merged. Two out of eighteen students were able to get a solo bug pull request merged, a remarkable accomplishment in a 10-week term.

3.1.4 *Student Observations.* Students were required to blog as well as to complete a reflection paper at the end of the term. In these, a large number of students report that they have learned much more than they expected. However, they also talk about the difficulty of jumping into the project and having to figure it out. Ultimately, the rewards of taking that risk are great.

Students reported on the advantages of working with a large code base "Looking at unfamiliar code, and professional code at that, has given me a better understanding of how to code and helped to better my own coding. Just working with the size of the devtools codebase has given me an appreciation for how large scale some of these projects can get, ... The diversity of people working on the project too has also made it less intimidating for me to get into as it shows me that any and all can help contribute to the project."

Additionally group work was discovered to be advantageous and community factored into success. One students reported "What I learned from this course is how beneficial working with groups and in a community can be …I always thought that there was this stigma that you had to become this master coder and be able to do anything and everything to do with computers by yourself. That if a hard problem had arisen, you and solely you had to figure out the solution. Participating in a HFOSS project showed me - physically in class and virtually on-line - that it isn't anything like that." Two other students commented on community "In the end I feel like I can understand the process and interactions inside the Open Source community much better than when I began. Locating a bug is much easier now, as well as how to get additional information from those who have been working within the project for a while. I am no longer nervous to dive into fixing a bug and I hope I can continue to find the time to contribute much more to Open Source. "In reference to taking on a second bug: "It also didn't take too long to figure out what I needed to do., because of how much I learned from my time on bug 1."

Students have noted that having the Mozilla community willing to help and respond quickly to questions in a kind and helpful way made a process they were initially intimidated by easier and it helped build their confidence. One student, reacting to the community, said, "I think it is worth mentioning that every one that I have personally communicated with has been extremely professional, kind and helpful. There is a positive attitude that is contagious while working with HFOSS, and everyone seems to be very encouraging of new members." While another student talked about overcoming fear, "Participating in large projects like this can be so intimidating but don't let that stop you. The community is so friendly and open to it. Just jump in and they'll help you in every way they can." It was also noted that members of the community were a big help in getting students involved, "What was your greatest source of help in becoming engaged? The member darkwing on GitHub. We communicated with him quite a bit and he was very helpful and quick to respond to questions. We definitely couldn't have completed our first bug if it wasn't for his help."

Additionally, as a teacher, approaching an inclusive community makes it easier to get questions answered and to find community members who are willing to help.

3.2 Drexel University

Drexel University has about 26,000 students and is located in Philadelphia, PA. The University has a strong focus on applied technology and is known for an emphasis on cooperative education. The College of Computing and Informatics has about 1,800 students including 1,200 undergraduates who all participate in co-operative education as part of their degree programs.

3.2.1 *Course Overview.* Humanitarian Open Source was offered as an 11 week special topics course in the spring quarter of 2018. There were 11 students in the class, primarily computer science majors in their last two years of study. The first part of the course included readings,

assignments, and some lectures to introduce students to open source. This included coverage of the history of open source, the difference between the Free Software movement and the Open Source Initiative, exploration of typical HFOSS project features, tools, and processes, and considerations for evaluating open source projects. The later part of the term focused on exploring and participating in HFOSS projects.

3.2.2 *Project.* The HFOSS projects for the term were limited to a set of projects related to food and nutrition. Students were given a short list of projects to explore and they self-organized around two projects: Open Food Facts, which is a crowd-sourced database of information on food products, and Pantry for Good, which is a food bank logistics and management project. Once projects were selected, students shared information about the project, but were free to work independently on aspects that interested them.

3.2.3 *Results.* Students explored the project that they selected starting with installing the development environment and reviewing features of the production environment. Students mostly chose to focus on known code defects, although a few spent time working on product documentation or language translation needs as well. Of the eleven students in the class, 8 had at least one contribution accepted by the project, and 6 of those had multiple contributions accepted. All of the students reported significant learning from their efforts, even when contributions were not accepted by the project.

3.2.4 *Student Observations.* Several themes emerge from reflective writing done by students during the term, including:

Subject matter knowledge - quizzes and student comments indicate that students entered the class with relatively little knowledge of many aspects of open source. This includes the legal mechanisms that enable open source, the size of the open source market, the role of community in governance, open source as a career, and open source business models. Students also reported gaining additional understanding of technical processes and tool use. For example, while all the students were familiar with GitHub and basic revision control operations, some of them had little exposure to how this translated into workflow for distributed development. One student commented: "I have always been comfortable with pushing and pulling in Git, but I never knew about pull requests or merging into major projects. Actually going through the process of having to do a pull request taught me so much about how to work on a major project with other contributors."

Community - Students also became much more aware of the importance of community in open source projects, and the value for new participants of connecting with the community. When asked what advice they would offer to other students interested in HFOSS participation, many of the students pointed to the importance of connecting with the community. One student suggested: "it's better to work within the community and not as an individual outsider", and another offered: "From my experience in starting out as a contributor, it is always a good first step to interact with current contributors to get a more detailed description of the history and the goals of the project." Students also noted that they found the project communities friendly and willing to help a new contributor. One student noted: "I was surprised at prompt and detailed replies to my questions, and felt very

welcome."

Starting Small - In looking at student success and struggles, the three students who did not have any contributions accepted by projects all spent considerable time learning new technologies in addition to learning about the HFOSS project. While all of the students learned some new technology, it seemed important to keep a balance between new technology and the time needed to understand the HFOSS project. Several of the successful students noted that they started with simple tasks. One student suggested that a new contributor should work on "the easiest and most straightforward issue they can find." Another suggested: "When contributing to a FOSS project for the first time, you should start off with something small."
Social Good - Finally, several of the students were quite interested in the humanitarian aspect of HFOSS. One of the women in the class noted: "I've always wanted to make and contribute to software that helps people in the world, not just to make a good salary, and work on HFOSS projects is a great way to do that!"

3.3 Western New England University

Western New England University (WNE) is a small (approx. 2500 undergraduates) liberal arts University located in Springfield, Massachusetts. The Computer Science degree is offered by the Department of Computer Science and Information Technology that is housed in the College of Arts \& Sciences. There are approximately 90 Computer Science (CS) majors and 35 Information Technology (IT) majors.

3.3.1 *Course Overview.* CS-490 Software Engineering is a senior-level course required for all CS majors. The course is taught in a 15-week term and is a typical project-based Software Engineering course offered in one term of the senior year of an undergraduate degree program. Emphasis is on the software development phases and an HFOSS project is used throughout the term to demonstrate concepts while providing hands-on project experience for students. There were 18 students in the fall 2017 offering of the class.

Assignments are a mix of project deliverables and homeworks. The project deliverables include a requirements document, requirements review document and design document. The individual homework assignments facilitate students becoming involved in the HFOSS project. There is a reflection paper due at the end of the term that requires students to reflect on their learning experience.

3.3.2 *Project.* The class focused on Accessibility aspects of the Mozilla Developer tools (dev tools) project. In particular, the class explored the accessibility issues related to the dev tools debugger. Students were split into four teams of either four or five students. There was one blind student in the class which provided that student's team a real-world user of the application. Each team identified a bug to work on related to allowing users with limited sight to use the Debugger.

3.3.3 *Results.* Students were able to construct Requirements, Design and Test documents and each team worked on an accessibility bug. However, no team got far enough to submit a pull request.

3.3.4 *Student Observations.* Students were required to submit a reflection paper at the end of the term that discussed what they learned and how their learning was different in the CS 490 Software Engineering course. This section presents a summary of student observations about the course and their learning.

Many students commented about the unstructured nature of the course and the impact of working with a real project with real problems. One student commented, "Working in a professional environment such as Mozilla is completely different from everything I have done prior to this class." Two students commented about the motivation for learning: "A good amount of learning is shifted out of the classroom and instead relies on the individual to figure things out." In addition, multiple students commented on the open- ended nature of the course with one student indicating "There was no 'solution' to our problem that we would simply need to implement. We needed to search the code in order to understand what parts were related to our bug, and which parts would be needed to be changed in order to fix the bug. This is very different from the way that a typical classroom operates, because usually there are assignments that are given which have a predetermined solution that you can work towards."

The difference in the course impacted students' learning style and enhanced critical thinking and problem-solving skills. One student indicated that "...the way we applied our knowledge was not the assumed accumulation of everything we had learned in computer science courses, instead it was how we were taught to think in our courses." while another stated "... we had to learn to figure out the answers, and the resources to those answers, on our own." The course appears to prepare students to enter the professional world with one student indicating, "The class experience was a reality check on how potential workplace environments will be. Tough and demanding, Software Engineering developed me into a more resourceful individual and showed me the different ways that the open source world can help me find answers once [I have] exhausted all other options." Another student stated "What counted was brute force determination, problem solving skills, and being willing to realize you were hopelessly lost without giving up entirely. There's a lot to learn, and a lot of experiences that are very worth having. They come at a price though, and that price, at least for me, was a lot of failures along the way."

Most students commented on the usefulness of the community in their learning. Students commented: "We were also taught, in more ways than one, by the Mozilla community.", "In a sense the main strength of a FOSS community is the community itself, and having this community to reach out to while working to help them in a class was an immense aid and a learning experience in and of itself." Another student commented on the value of working in a professional community, "...how often are students given the opportunity to work on projects with non-students?" The Mozilla Dev Tools community was very supportive of students with one student commenting, "Never once did the community make me feel like I was a nuisance to them, as some of them almost seemed eager to help. It was very refreshing and almost motivating to have the community supporting us through it and essentially being a pseudo-professor for all of us."

Students also appeared to gain valuable teamwork skills and to appreciate working in teams. One student noted, "In a classroom setting, talking to someone outside of class to figure out how to do something or how to get an answer would be considered cheating. However, in the FOSS environment, talking to others is encouraged and often times the only way of solving a problem."

Several students commented on the size and complexity of the project and the impact on their learning. One student indicated "Learning that even small projects can take weeks to produce even a few lines of code was astounding to me." Others commented on the size of the code base "There are tens of thousands of lines of code compared to the in-class assignments' usual around one-hundred lines of code."

Throughout the papers, there was a recognition that students were learning about how to be a professional software engineer with one student commenting "It is common for classes to teach you the content of what you need to be successful in a certain field, but not that common to teach students about how that field works on a day to day basis." Students also commented on gaining communication skills with one student stating "Through this we learned how to communicate efficiently and effectively to work on a small part of a large scale project which is something we will be doing in the real world every day." Another commented about the impact of communication on other professional skills "This communication was also a helpful learning tool as it taught us to think critically about the format and professionalism of our communication in a large community space, and essential skill in real world on the job communication."

Perhaps most importantly, many students commented on their learning within the course. Students indicated learning about development skills with one student commenting: "the experience made me integrate some better practices into how I write code, which I hope will have a positive effect on my skills and will make me more prepared for working as a full-time developer." Several students commented that they gained a better understanding of software engineering as a discipline with one student stating: "I feel as though this class has taught me beyond the overall scope of a general class. I am confident that with this course I have excelled in determining what it really means to be a software engineer." and another saying: "…it felt like we had worked on and understood the process of a real software project from start to finish, which was the main goal of the class." Lastly, students appeared to grow in professional skills with a student stating: "I grew as a developer, a person, and as a communicator."

It should be noted that many students indicated a sense of frustration or of being overwhelmed. Others commented on issues such as lack of documentation, delayed community response time, and size of the project. However several of these same students commented that they were glad that they had encountered this while in a classroom setting rather than in their first professional position.

4 DISCUSSION

This section summarizes some of the common themes that emerge in looking across the three courses and considering student observations.

- Knowledge and Skills - Many student comment on HFOSS participation as a "real-world" professional experience. This includes comments about the chance to integrate various aspects of software engineering process, dealing with large code bases, developing new appreciation for team processes and communication. Students understand that this is not just textbook learning, but that they are observing and participating in actual project work. Many student comments about learning focus on integration of prior learning and learning about professional practice. It is this sort of learning that seems to separate involvement in an HFOSS project from typical course work. This difference also seems to heighten student engagement considerably.
- Motivation and Confidence - Students in these classes are in the final years of their undergraduate degree, and generally settled on a computing major, so there is little indication that HFOSS participation affects choice of major. On the other hand, there are many aspects of student comments that indicate students are motivated by HFOSS participation, and develop confidence in their ability to interact with computing professionals. Students often indicate that the course has increased their understanding of their chosen profession.
- Community Interaction - Interaction with the HFOSS project community is central to student learning. Students generally report supportive interactions with HFOSS project communities. While this sort of reception is not guaranteed, students do seem to benefit from and be energized by interaction with community members. Being inclusive and welcoming is not always the reputation of open source projects, so this experience may be related to the humanitarian nature of the projects selected for these classes. Students that engaged the HFOSS community early tend to have a better understanding of the project's requirements and of the project overall. They were more successful and generally seemed to find the experience more rewarding. The experience gave students a different sense of FOSS community dynamics and the important role of communication in distributed development. Multiple comments described a transition by students from being reluctant to engage with the community to endorsing early and frequent interaction.
- Student Contributions - There are a variety of contributions that students can make and taking advantage of these opportunities when possible can aid student learning. Known defects are often a good opportunity for student contributions. However, work with documentation or language translation are valued contributions as well. Code and processes are frequently undocumented. Having students document code is an excellent way for them to learn about the code base. For an instructor, it helps to design a course such that different students may make different contributions within the course structure.

4.1 Suggestions for Adoption

Instructors who are interested in exploring student participation in HFOSS projects might consider the following points:

- Start Small - The easiest way to introduce HFOSS into a curriculum is to start small. It is easier to get a grasp on how to work with HFOSS as a professor by finding one or two assignments for a class. Alternatively, HFOSS participation can be introduced by offering an independent study course for one or more students. These initial experiences can provide a basis for more extensive work in HFOSS with students.
- Utilize the Open Source Community - The collective understanding and knowledge of the community supporting an HFOSS project is far beyond that of an individual instructor. While the instructor must provide milestones and grading, the project community can be a resource for answering questions and helping to guide students.
- Consider Extracurricular Options - In some circumstances the best option for student learning by HFOSS participation may be via extracurricular activities. This could be as part of the programming of an existing student club, or a new club focused on open source. The Mozilla Foundation has an initiative to help student open source clubs succeed. (See https://opensource.mozilla.community/)
- Connect with other Instructors - TeachingOpenSource.org is an online community that has a number of learning activities designed for all levels of a computer science curriculum. All these materials are available under a Creative Commons license. Having other professors who are teaching using HFOSS to get advice from on what projects to consider or how to approach a community, to get assignments, or figure out how to assess participation in HFOSS is important. POSSE, the Professors' Open Source Software Experience, also offers a workshop to help learn how to teach students how to participate in HFOSS. (See http://foss2serve.org/index.php/POSSE)

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant Nos. - DUE-1225708, DUE-1225738, DUE-1225688, DUE-1525039 DUE-1524898, and DUE-1524877. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

REFERENCES

[1] 2016. 10th Anniversary of the Open Source Survey. (2016). www.northbridge. com/2016-future-open-source-survey-results

[2] S. Beyer, K. Rynes, and S. Haller. 2004. Deterrents to women taking computer science courses. *IEEE Technology and Society Magazine* 23, 1 (Spring 2004), 21–28. DOI:http://dx.doi.org/10.1109/MTAS.2004.1273468

[3] Grant Braught, John Maccormick, James Bowring, Quinn Burke, Barbara Cutler, David Goldschmidt, Mukkai Krishnamoorthy, Wesley Turner, Steven Huss-Lederman, Bonnie Mackellar, and Allen Tucker. 2018. A Multi-Institutional Perspective on H/FOSS Projects in the Computing Curriculum. *ACM Trans. Comput. Educ.* 18, 2, Article 7 (July 2018), 31 pages. DOI:http://dx.doi.org/10.1145/3145476

[4] Bo Brinkman and Amanda Diekman. 2016. Applying the Communal Goal Congruity Perspective to Enhance Diversity and Inclusion in Undergraduate Computing Degrees. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 102–107. DOI: http://dx.doi.org/10.1145/2839509.2844562

[5] A.J. Budd and H. J. C. Ellis. 2008. Spanning the gap between software engineering instructor and

student. In *2008 38th Annual Frontiers in Education Conference*. S3H–10–S3H–15. DOI:http://dx.doi.org/10.1109/FIE.2008.4720516

[6] Kevin Buffardi. 2017. Comparing Remote and Co-located Interaction in Free and Open Source Software Engineering Projects. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA, 22–27. DOI:http://dx.doi.org/10.1145/3059009. 3059019

[7] Lori Carter. 2006. Why Students with an Apparent Aptitude for Computer Science Don'T Choose to Major in Computer Science. *SIGCSE Bull.* 38, 1 (March 2006), 27–31. DOI:http://dx.doi.org/10.1145/1124706.1121352

[8] J. McGrath Cohoon. 2002. Recruiting and Retaining Women in Undergraduate Computing Majors. *SIGCSE Bull.* 34, 2 (June 2002), 48–52. DOI:http://dx.doi.org/ 10.1145/543812.543829

[9] Allan Collins. 2006. Cognitive apprenticeship: The cambridge handbook of the learning sciences, R. Keith Sawyer. (2006).

[10] Michelle Craig, Ted Kirkpatrick, Shealen Clare, and Amgine Saewyc. 2012. Undergraduate Capstone Open-source Projects. In Proceedings of the Seventeenth Western Canadian Conference on Computing Education (WCCCE '12). ACM, New York, NY, USA, 57–58. DOI:http://dx.doi.org/10.1145/2247569.2247589

[11] T de Lanerolle, Ralph Morelli, Norman Danner, Danny Krizanc, Gary Parker, and Ozgur Izmirli. 2008. Creating an academic community to build humanitarian foss: A progress report. In *Proceedings of the 5th International ISCRAM Conference*. 337–341.

[12] Diana Drechsel. 2018. Research-based Gender Competences As a Professional Skill in STEM Exemplified by the "fix-IT. Fixing IT for Women" Project. In *Proceedings of the 4th Conference on Gender & IT (GenderIT '18)*. ACM, New York,NY, USA, 147–151. DOI:http://dx.doi.org/10.1145/3196839.3196862

[13] Heidi J.C. Ellis, Gregory W. Hislop, Monisha S. Pulimood, Becka Morgan, and Ben Coleman. 2015. Software Engineering Learning in HFOSS: A Multi-Institutional Study. *122nd Association for Engineering Education Annual Conference and Exposition* (2015).

[14] Heidi J.C. Ellis, Gregory W. Hislop, Josephine Rodriguez, and Ralph A. Morelli. 2012. Student Software Engineering Learning via Participation in Humanitarian FOSS Projects. *119th Annual ASEE Conference and Exhibition* (2012).

[15] Heidi J. C. Ellis, Gregory W. Hislop, Mel Chua, and Sebastian Dziallas. 2011. How to Involve Students in FOSS Projects. In *Proceedings of the 2011 Frontiers in Education Conference (FIE '11)*. IEEE Computer Society, Washington, DC, USA, T1H–1–1–T1H–6. DOI:http://dx.doi.org/10.1109/FIE.2011.6142994

[16] Heidi J. C. Ellis, Gregory W. Hislop, Stoney Jackson, and Lori Postner. 2015. Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS). *Trans. Comput. Educ.* 15, 4, Article 18 (Dec. 2015), 23 pages. DOI: http://dx.doi.org/10.1145/2684812

[17] Heidi J. C. Ellis, Gregory W. Hislop, Michelle Purcell, and Lori Postner. 2013. Project Selection for Student Participation in Humanitarian FOSS. *J. Comput. Sci. Coll.* 28, 6 (June 2013), 16–18. http://dl.acm.org/citation.cfm?id=2460156.2460162

[18] Heidi J. C. Ellis, Stoney Jackson, Gregory W. Hislop, Lori Postner, and Darci Burdge. 2014. Getting Started in Open Source a Tour of a Real Project. *J. Comput. Sci. Coll.* 29, 6 (June 2014), 12–14. http://dl.acm.org/citation.cfm?id=2602724. 2602728

[19] Rüdiger Glott, Andreas Meiszner SPI, Sulayman K Sowe, Thomas Conolly, Ashley Healy, Rishab Ghosh, Athanasis Karoulis, Hugo Magalhães SPI, Ioannis Stamelos, Martin J Weller, and others. 2011. FLOSSCom-Using the Principles of Informal Learning Environments of FLOSS Communities to Improve ICT Supported Formal Education. (2011).

[20] Swapna S. Gokhale, Thérèse Smith, and Robert McCartney. 2012. Integrating Open Source Software into Software Engineering Curriculum: Challenges in Selecting Projects. In *Proceedings of the First International Workshop on Software Engineering Education Based on Real-World Experiences*

(EduRex '12). IEEE Press,Piscataway, NJ, USA, 9–12. http://dl.acm.org/citation.cfm?id=2663678.2663681

[21] Gregory W. Hislop, Heidi J.C. Ellis, and Ralph A. Morelli. 2009. Evaluating Student Experiences in Developing Software for Humanity. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09)*. ACM, New York, NY, USA, 263–267. DOI:http://dx.doi.org/10.1145/1562877.1562959

[22] Gregory W. Hislop, Heidi J.C. Ellis, S. Monisha Pulimood, Becka Morgan, Suzanne Mello-Stark, Ben Coleman, and Cam Macdonell. 2015. A Multi-Institutional Study of Learning via Student Involvement in Humanitarian Free and Open Source Software Projects. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15)*. ACM, New York, NY, USA, 199–206. DOI:http://dx.doi.org/10.1145/2787622.2787726

[23] Rick Homkes. 2008. Assessing It Service-learning. In *Proceedings of the 9th ACM SIGITE Conference on Information Technology Education (SIGITE '08)*. ACM, New York, NY, USA, 17–22. DOI:http://dx.doi.org/10.1145/1414558.1414564

[24] Nazish Zaman Khan and Andrew Luxton-Reilly. 2016. Is Computing for Social Good the Solution to Closing the Gender Gap in Computer Science?. In *Proceedings of the Australasian Computer Science Week Multiconference (ACSW '16)*. ACM, New York, NY, USA, Article 17, 5 pages. DOI:http://dx.doi.org/10.1145/ 2843043.2843069

[25] Clif Kussmaul, Heidi J.C. Ellis, and Gregory W. Hislop. 2012. 50 Ways to Be a FOSSer: Simple Ways to Involve Students & Faculty (Abstract Only). In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. ACM, New York, NY, USA, 671–671. DOI:http://dx.doi.org/10.1145/ 2157136.2157393

[26] Chang Liu. 2005. Enriching Software Engineering Courses with Service-learning Projects and the Open-source Approach. In *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*. ACM, New York, NY, USA, 613–614. DOI:http://dx.doi.org/10.1145/1062455.1062566

[27] Robert Marmorstein. 2011. Open Source Contribution As an Effective Software Engineering Class Project. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education (ITiCSE '11)*. ACM, New York, NY, USA, 268–272. DOI:http://dx.doi.org/10.1145/1999747.1999823

[28] Andreas Meiszner, Rüdiger Glott, and Sulayman K Sowe. 2008. Free/Libre Open Source Software (FLOSS) communities as an example of successful open participatory learning ecosystems. *UPGRADE, The European Journal for the Informatics Professional* 9, 3 (2008), 62–68.

[29] Ralph Morelli, Allen Tucker, Norman Danner, Trishan R. De Lanerolle, Heidi J. C. Ellis, Ozgur Izmirli, Danny Krizanc, and Gary Parker. 2009. Revitalizing Computing Education Through Free and Open Source Software for Humanity. *Commun. ACM* 52, 8 (Aug. 2009), 67–75. DOI:http://dx.doi.org/10.1145/1536616. 1536635

[30] Becka Morgan and Carlos Jensen. 2014. Lessons Learned from Teaching Open Source Software Development. In *Open Source Software: Mobile Open Source Technologies*, Luis Corral, Alberto Sillitti, Giancarlo Succi, Jelena Vlasenko, and Anthony I. Wasserman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 133–142.

[31] 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 *Frontiers in Education Conference, 2013 IEEE*. IEEE, 1837–1843.

[32] Lori Postner, Heidi J.C. Ellis, and Gregory W. Hislop. 2018. A Survey of Instructors' Experiences Supporting StudentLearning using HFOSS Projects. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (SIGCSE '18). ACM, New York, NY, USA, 203-208. DOI: https://doi.org/10.1145/3159450.3159524

[33] Eric Raymond. 1999. The cathedral and the bazaar. *Knowledge, Technology & Policy* 12, 3 (1999), 23–49.

[34] Jane Stout and Burçin Tamer. 2016. Collaborative Learning Eliminates the Negative Impact of Gender Stereotypes on Women's Self-Concept (Abstract Only). In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 496–496. DOI:http://dx.doi. org/10.1145/2839509.2850515

[35] Etienne Wenger. 1999. *Communities of practice: Learning, meaning, and identity*. Cambridge university press.