Scaffolding Student Success in the Wilds of Open Source Contribution

Emily Lovell
Computer Science Department
UC Santa Cruz / Berea College
Santa Cruz, CA / Berea, KY
emme@soe.ucsc.edu

James Davis
Computer Science Department
UC Santa Cruz
Santa Cruz, CA
davis@soe.ucsc.edu

***Abstract*—This Innovative Practice Work in Progress paper reports on our experience scaffolding student success in the uncertain landscape of open source. Following participation in a faculty workshop on the subject, the first author spent two consecutive terms developing, teaching, and revising an upper-division open source software course. The difference between the two course offerings was astounding; students enrolled in the second iteration made more successful project contributions, spent more of their own time working outside of class, and felt a greater connection to both the project and the developer community of which they were a part. We detail our experiences here, with particular focus on the importance of project selection – as well as the revisions we believe to be most responsible for improvement: additional mentorship, supplemental in-class tutorials, more dedicated class time for teamwork, intentional team groupings, and access to large screens for collaboration.**

***Keywords*—Professional skills, mentoring, computer science, team based learning, higher education, student diversity, under-representation, course design, instructional design, recruitment and retention, communication skills, active learning**

I. Introduction

Teaching open source software development has gained traction in undergraduate curricula for many reasons: students learn to use real-world tools/processes, build portfolios of project contributions, and function within a distributed professional community [1], [2]. Open source also provides a clear avenue for students to have a positive and tangible impact on society, something that is known to be relevant to broadening participation in computing [3], [4].

Open source contribution also showcases an application of computer science that students may not have been aware of prior. At the upper-division course level, this is especially important, as students are about to decide whether they will seek a career in computing post-graduation [5].

We became interested in teaching open source after one of the authors volunteered at a workshop through OpenHatch's *Open Source Comes to Campus* program [6]. The workshop was held at a different diverse, minority-serving institution than her own – and she observed how motivating it was for these students to learn real-world tools and contribute to a real-world project. Soon after, we discovered Foss2serve [7], an established community of educators and researchers supporting student involvement in humanitarian open source projects. We also learned of *POSSE* (the Professors' Open Source Software Experience), which is a professional development workshop offered by RedHat and Foss2serve to support faculty new to teaching open source.

The first author participated in two *POSSE* workshops, during which she gained experience with open source tools and helped to develop curriculum and activities for college classroom use [8], [9]. She learned from faculty at a variety of institutions who were integrating open source into their computer science courses and began envisioning how she might teach such a course of her own. Upon starting her first tenure-track position, she was invited to do exactly this.

Over the course of two consecutive terms, the first author developed, taught, and heavily revised an upper-division course entitled *Open Source Software Engineering*. She leveraged a wealth of existing activities and resources in designing the course [7], was supported with real-time mentorship as she ran the course (from other faculty involved with POSSE), and secured sustained professional mentorship for her students (from the Mozilla DevTools project).

The difference between the two terms was tremendous. Most notably, student project contributions increased from a 25% success rate in the first iteration to 100% in the second iteration. As a result of the course's success, it was added to the department's permanent course catalog. In this paper, we detail our experiences across these two terms and the revisions that we believe to be responsible for the improved student experience and outcomes in the second iteration.

II. Related Work

There exists a growing body of research on undergraduate engagement in free and open source (FOSS) projects, especially humanitarian free and open source projects (HFOSS). It has been well established that involving students in open source communities offers a valuable opportunity for students to learn within a community of practice, gain experience with practical tools (e.g. version control systems, bug trackers), build a portfolio, and contribute to a real-world project [10], [11], [12]. Contributing to an open source project also helps students cultivate "soft" skills such as public speaking, project management, delegation, and working with a team.

HFOSS projects, in particular, have been a deliberate choice for many educators because these communities are typically

welcoming and supportive to newcomers [13]. It is suspected that these communities also attract participation from women and other underrepresented minorities, due to their social impact [14]. Additionally, the altruistic nature of humanitarian open source contribution lends itself to service-learning [15]. Teaching open source also offers instructors the chance to model a growth mindset and to foster a sense of belonging within a professional community – also of great relevance to broadening participation [16], [17], [18].

Teaching open source, however, presents many curricular challenges: community leadership can take unexpected turns, projects vary in size and complexity, and student learning can be difficult to assess [1]. Our work puts the aforementioned research into practice and reports on what we have learned; scaffolding student learning in such an unpredictable context is challenging, but thoughtful planning and revision can have a dramatic positive impact on outcomes.

III. Course Overview

Although our course is titled *Open Source Software Engineering*, the emphasis is much more on open source than on software engineering practices. The first half of each term is spent on history, etiquette, culture, and tools – and the second half is spent diving into an active open source project.

The first offering consisted of 16 students (14 male, 2 female) and the second offering consisted of 12 students (10 male, 2 female). Both classes represented a wide range of experience, as some students had only taken CS1 and CS2 while others had completed a variety of upper-division coursework. An overview of the two offerings appears below.

A. Term 1

We leveraged the Foss2serve library of activities for the first half of the course, guiding students through licensing, candidate project evaluation, version control with git, communication tools (such as IRC and Slack) and more. Students completed these exercises in pairs, in class. We also required students to make GitHub accounts and, after an in-class crash-course on HTML and CSS, students practiced fixing up a buggy GitHub Pages site that was created in advance. Several GitHub "issues" were generated for students to claim and work on, to help learn the GitHub workflow and to practice HTML/CSS. (This activity was borrowed from OpenHatch.)

Weekly reading was assigned from either *The Cathedral and the Bazaar* [19] or *The Art of Community* [20]. Students were required to create blogs and to post reading responses there. A few group discussions were held in class on related topics.

For the second half of the term, we embedded all students in the same open source project rather than each team selecting their own project. Students weren't very excited about this approach, but we felt that it would be easier for us to support them – and for them to support one another. We solicited suggestions through the Teaching Open Source [21] mailing list and learned of others' positive experience engaging with the Mozilla Firefox DevTools community [22] and, more specifically, with the debugger.html project [23]. We connected with two other faculty members teaching with DevTools and together we worked with the debugger.html community to identify candidate bugs for our students. Motivated by a desire to engage students in HFOSS, we selected bugs under the umbrella of accessibility.

Students were divided into teams of four. We grouped students according to their own preferences and who we thought would work well together. One consequence, however, was that most teams reflected a broad spectrum of prior experience. We instructed students to assign themselves relevant homework between class meetings – e.g. tutorials, testing, or bug research – and asked that they reserve class time for team collaboration. Students completed bi-weekly team evaluations, in which they ranked themselves and each of their teammates on metrics like regular attendance, leadership, and attitude. (This tool was shared with us by a faculty collaborator.) These were treated as confidential and allowed a window into any interpersonal challenges early enough to intervene.

Students were required to join the *Open Source Software Engineering* channel on the department's Slack team, which was created and is managed by teaching assistants (TAs). Each team was also asked to create *their own* Slack channel, in which they would briefly report out in writing at the start and finish of each class. This was inspired by scrum/standup meetings; each student had to share what they accomplished outside of class and what they would spend class time working on that day. This gave students experience with industry practices and tools and also helped guide the instructor as to which teams needed help getting unstuck. This practice also held students accountable, as teammates would be disappointed if someone had not done any work between course meetings.

During class, teams worked – sometimes altogether, sometimes in pairs – to make progress on their chosen bug. This often involved posting to communication channels used by the debugger.html project, including Slack and GitHub. Through our community interactions, it became apparent who a couple of particularly helpful Mozilla developers were, and we leveraged their support through the rest of the term.

The debugger.html project is built in React [24], which is not covered anywhere in our departmental curriculum. No structured support was provided for learning React; instead, each team sought out materials to learn the basics, and sometimes students shared resources across teams.

Three out of four teams got so far as to submit pull requests on GitHub. However, only one team's contribution was accepted and merged. The other teams got stuck in the review process – or in one case, were unable to even fully solve/address the bug they had been working on all term.

B. Term 2

We made significant changes to the course, both in response to student feedback and our own observations. We also hired a TA who had previously taken the class and was able to provide feedback based on his experience as a student in the course. Below, we summarize the major changes.

Students reported getting little out of the readings from *The Art of Community*, so we dropped that textbook. Because discussions had been sparsely participated in, we spent less class time on them and instead asked students to reflect deeper in their blog posts. These changes won more class time for working in pairs or teams.

Students from the prior term struggled with learning React and expressed frustration with each team discovering the same resources on their own. In response, the instructor asked the course TA to develop and lead a walkthrough in which students built a barebones blog using React. We also created a shared virtual bulletin board (using Trello [25]), where students posted resources that their classmates might find useful.

We chose to involve students in the same project as before: debugger.html. This time, students were grouped with those of *similar experience level*, allowing each team to choose an appropriately challenging bug to tackle. Students were also grouped in *teams of three instead of four*, as we suspected this might lead to more consistent communication within teams. Teams did collaborate more effectively this way, with each member contributing more equally to conversations and to code.

The Mozilla developers that we encountered in Term 1 brainstormed with the instructor about how to better support students through the contribution process. We decided to identify smaller issues – or even subtasks of issues – for teams to claim, even if it meant shifting focus beyond accessibility. We also established a separate Slack channel on the DevTools team, which both the students and the developers joined. This offered a less intimidating venue for students to ask questions. Students were required to cross-post their scrum reports in this channel, so that the developers could track their progress in greater detail. We believe that this gave students' self-assigned homework a greater weight, as they were reporting to real-world developers, and not just to their college instructor and classmates. We also added in-class standup meetings on a weekly basis, in which teams would report out to one another on their progress and share learned expertise.

The teams of three grappled initially with how to collaborate during class; no longer could they divide-and-conquer by splitting into pairs. They began making use of large portable screens in the classroom, which they wheeled to their desk clusters and took turns connecting their laptops to. This facilitated much richer discussion about each team's progress, as teams could analyze code, write code, or sift through resources together. Instead of watching pairs head-down at their laptops, we saw teams engaging in lively discussion, moving around and using the screen as a prop. This also made it easier for the instructor to circulate throughout the classroom and monitor each team's progress, joining their discussions when helpful.

Towards the end of the course, the Mozilla developers who were supporting our students offered to schedule a video call during class time. We structured this call as a standup meeting, during which each team reported out on their weekly progress and had the opportunity to receive real-time feedback. Students were also able to ask the developers about their personal experience getting into open source.

Each of the four teams made at least one successful contribution to the project. One team made three, spanning both code and documentation!

IV. Student Feedback

We asked for informal feedback throughout both terms in which the course was offered. In Term 1, students expressed a large degree of frustration and confusion (although they responded to it with a constructive attitude), while students in the second term openly and enthusiastically affirmed that they were having a positive learning experience. Unsolicited, the instructor received the following from a student via email, about halfway through Term 2:

> "So far I am genuinely enjoying the course. The work is not too overwhelming and it feels manageable. I really like how we are encouraged to try things and learn on our own. It is building my confidence as a woman in computer science."

We also received valuable feedback on the course through students' course evaluations, submitted at the end of each term. A couple of comments following Term 2:

> "I did learn a lot. I feel much more comfortable with my computer, with web development, with open source, with communicating, with teamwork, and everything we touched on in class."

> "I learned more about open source development than I even expected to in this course. I think the idea of having students contribute to a real piece of software is amazing and it is a piece of software that millions of people, including myself use. Interacting with the Firefox community was very educational both in a coding aspect and in a... well, community aspect."

Comparing quantitative evaluation data, students from Term 2 spent more hours per week on the course, reported learning more, and rated the course higher overall.

V. Reflections & Recommendations

It may seem obvious that a course should improve in its second offering, due to the instructor's increased familiarity with the material/structure and access to an experienced TA. However, this course improved dramatically, and despite a continually shifting context. Below, we summarize what we believe to be the most influential factors.

A. Project Choice

Project choice is arguably the most foundational factor in the success of any class structured around open source contribution. We recommend, when possible, embedding all students in one project/community; this allowed the instructor to understand and support student progress while also staying in touch with a single set of community leaders. As is emphasized by Foss2serve, we also recommend verifying that the community is highly active and welcoming to newcomers; this

will ensure that students receive timely, constructive responses to questions and pull requests. In our case, debugger.html's active Slack channel also meant that students could observe community norms before wading in themselves.

B. Team Formation & Tools to Support Collaboration

After attempting two different strategies for assigning teams, we feel strongly that it's best to group students with others of similar experience level. This way, less experienced students do not fall behind or lose confidence – while more experienced students can take off and run with a more difficult problem.

As time permits, the more experienced teams can also provide support to those that are stuck. This is easily facilitated by the addition of class-wide standup meetings, during which stalled teams can solicit help. We also recommend team-specific Slack channels for communication/reporting, along with a class-wide channel for students to ask questions and share resources between class meetings.

Finally, a team size of three – along with access to large shared screens – encourages lively discussion and equitable collaboration within each team.

C. Structuring Unfamiliar Tools & Technologies

Although having to learn new technologies – React, in this case – was not the insurmountable obstacle we expected it to be, it really helped to provide some structure around this in the second term. We recommend providing infrastructure for students to share resources, as we did with Slack and Trello. We also recommend offering project-specific demos and/or walkthroughs for students to build experience with any required tools or technologies. We believe that these things empowered students to more efficiently and confidently jump into working on their bug/issue. (Note that this did not deprive students of the opportunity to feel "productively lost"; there was still plenty of independent learning to be done!)

D. Professional Mentorship

Professional mentorship was a vital thread running through the entire course experience. We believe this to be, perhaps, the most influential factor in the course's improvement. When the Mozilla developers became more involved in Term 2, students responded with greater motivation and a stronger sense of accountability. Although many students initially found it intimidating to communicate directly with the developers, doing so pushed them to practice communicating with professionalism and specificity. These developers modeled a growth mindset; not always having the answers, but coaching students through finding resources and learning on the fly. They provided continuous and timely feedback via Slack and GitHub, doing so with proficiency, patience, and encouragement.

Students responded especially positively to the standup video call that we did towards the end of Term 2. Knowing the call was on the horizon motivated them to make progress as a team and to generate interesting questions. Additionally, they valued seeing that the developers were people that they could relate to; approachable individuals who once had very little experience with open source themselves.

The instructor also benefitted from mentorship – both from the developers (with whom she could check in about student progress and impact on their community) and from other faculty teaching open source (who offered mutual support and years of experience). This mentorship helped *her* to maintain a growth mindset as she guided students through unfamiliar content and processes.

For the above reasons, we emphatically recommend reaching out to others teaching with open source, as well as securing mentorship for your students within your chosen project.

VI. Future Work

Although we are confident that students learned more in the second offering of the course, assessing actual student learning in open source is challenging. In the case of this course, each student entered with a different level of experience, and it was important to us that students make progress relative to their own starting points. Blogs allowed a window into each student's process, but students did not seem motivated to complete these assignments thoughtfully nor in a timely fashion. Students echoed these sentiments in their course evaluations, along with an explicit desire to be assessed on their technical contributions. Adding an assessment of students' concrete technical contributions would serve as a good motivator in future offerings.

Given the positive response from our single standup scrum video call in Term 2, we believe scheduling those meetings more frequently would be beneficial – and the developers volunteered to do so in a future course offering.

Finally, a term-length course is barely long enough for students to dip their toes into an open source project. Many students have indicated interest in continuing their involvement in debugger.html and the instructor will be advocating for this to become an option for satisfying her department's senior project requirement.

VII. Conclusion

Open source is an uncertain and constantly shifting landscape within which to situate an undergraduate class – but the benefits are vast when well-executed. It is tricky to get right; despite participation in professional development workshops and connection with more experienced colleagues, students in the course's first offering were less successful than we would have preferred. The second offering resulted in substantially higher student success and we believe these gains were attributable primarily to thoughtful team formation, structuring unfamiliar tools and technologies, and professional mentorship. We hope that these findings are of use to others setting out to teach similar courses.

Acknowledgments

We thank the Mozilla DevTools debugger.html community, especially David Walsh and Jason Laster, for welcoming and supporting students across both course offerings. We also thank Foss2serve, especially Heidi Ellis and Darci Burdge, for their faculty mentorship on this project.

REFERENCES

[1] H. J. C. Ellis, G. W. Hislop, S. Jackson, and L. Postner, "Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS)," *ACM Transactions on Computing Education*, vol. 15, no. 4, pp. 18:1–18:23, Dec. 2015.

[2] G. W. Hislop, H. J. Ellis, and R. A. Morelli, "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*, ser. ITiCSE '09. New York, NY, USA: Association for Computing Machinery, Jul. 2009, pp. 263–267.

[3] J. Margolis and A. Fisher, *Unlocking the Clubhouse: Women in Computing*. MIT Press, Feb. 2003.

[4] A. B. Diekman, E. R. Brown, A. M. Johnston, and E. K. Clark, "Seeking congruity between goals and roles: A new look at why women opt out of science, technology, engineering, and mathematics careers," *Psychological Science*, vol. 21, no. 8, pp. 1051–1057, 2010, publisher: Sage Publications Sage CA: Los Angeles, CA.

[5] S. Yardi and A. Bruckman, "What is computing? Bridging the gap between teenagers' perceptions and graduate students' experiences," in *Proceedings of the Third International Workshop on Computing Education Research*, 2007, pp. 39–50.

[6] OpenHatch, "Open Source Comes to Campus." [Online]. Available: https://campus.openhatch.org/

[7] "Foss2Serve." [Online]. Available: http://foss2serve.org/index.php/Main_Page

[8] H. J. Ellis, M. Chua, G. W. Hislop, M. Purcell, and S. Dziallas, "Towards a model of faculty development for FOSS in education," in *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2013, pp. 269–273.

[9] B. Morgan, G. W. Hislop, and H. J. Ellis, "Faculty Development for FLOSS Education," in *IFIP International Conference on Open Source Systems*. Springer, 2019, pp. 165–171.

[10] H. J. Ellis, S. Jackson, D. Burdge, L. Postner, G. W. Hislop, and J. Diggs, "Learning within a professional environment: shared ownership of an HFOSS project," in *Proceedings of the 15th Annual Conference on Information Technology Education*, ser. SIGITE '14. New York, NY, USA: Association for Computing Machinery, Oct. 2014, pp. 95–100.

[11] H. J. Ellis, M. Chua, M. C. Jadud, and G. W. Hislop, "Learning through open source participation," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 2011, pp. 83–84.

[12] G. W. Hislop, H. J. Ellis, A. B. Tucker, and S. Dexter, "Using open source software to engage students in computer science education," in *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, 2009, pp. 134–135.

[13] G. W. Hislop, H. J. Ellis, S. M. Pulimood, B. Morgan, S. Mello-Stark, B. Coleman, and C. Macdonell, "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*, ser. ICER '15. New York, NY, USA: Association for Computing Machinery, Aug. 2015, pp. 199–206.

[14] L. Postner, D. Burdge, S. Jackson, H. Ellis, G. Hislop, and S. Goggins, "Using humanitarian free and open source software (HFOSS) to introduce computing for the social good," *ACM SIGCAS Computers and Society*, vol. 45, no. 2, pp. 35–35, 2015, publisher: ACM New York, NY, USA.

[15] R. Morelli, T. de Lanerolle, and A. Tucker, "The Humanitarian Free and Open-Source Software Project: Engaging Students in Service-Learning through Building Software," *Service-Learning in the Computer and Information Sciences: Practical Applications in Engineering Education*, pp. 117–136, 2012, publisher: Wiley Online Library.

[16] C. Lewis, P. Bruno, J. Raygoza, and J. Wang, "Alignment of goals and perceptions of computing predicts students' sense of belonging in computing," in *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 2019, pp. 11–19.

[17] N. Veilleux, R. Bates, C. Allendoerfer, D. Jones, J. Crawford, and T. Floyd Smith, "The relationship between belonging and ability in computer science," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 2013, pp. 65–70.

[18] E. Höhne and L. Zander, "Belonging uncertainty as predictor of dropout intentions among first-semester students of the computer sciences," *Zeitschrift für Erziehungswissenschaft*, vol. 22, no. 5, pp. 1099–1119, 2019, publisher: Springer.

[19] E. S. Raymond, *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary*, rev. ed. Beijing ; Cambridge, Mass: O'Reilly, 2001.

[20] J. Bacon, *The art of community: Building the new age of participation*. " O'Reilly Media, Inc.", 2012.

[21] "TeachingOpenSource – Instructors and open source communities supporting teaching open source." [Online]. Available: http://teachingopensource.org/

[22] "Firefox Developer Tools | MDN." [Online]. Available: https://developer.mozilla.org/en-US/docs/Tools

[23] "Introducing debugger.html – Mozilla Hacks - the Web developer blog." [Online]. Available: https://hacks.mozilla.org/2016/09/introducing-debugger-html

[24] "React – A JavaScript library for building user interfaces." [Online]. Available: https://reactjs.org/

[25] "Trello." [Online]. Available: https://trello.com/