# Using Scaffolding to Simplify FOSS Adoption

Stan Kurkovsky
Central Connecticut State University
New Britain, CT, USA
kurkovsky@ccsu.edu

## ABSTRACT

Free and open source software (FOSS) have become a popular framework for introducing students to professional software development. Despite a great variety of advantages, there is a significant number of barriers to a broader adoption of FOSS projects in computing education. We present a number of techniques forming a scaffolded agile approach to externally-sourced software engineering projects that help address some of the barriers to FOSS adoption.

## CCS CONCEPTS

• **Software and its engineering → Open source model**.

## KEYWORDS

Software engineering, FOSS, student projects

**ACM Reference Format:**
Stan Kurkovsky. 2022. Using Scaffolding to Simplify FOSS Adoption. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 2 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3502717.3532163

## 1 INTRODUCTION

In recent years, the use of free and open source software (FOSS) projects gained a significant momentum [1, 2]. FOSS provides students with an opportunity to participate in large-scale projects where they can apply their knowledge to solve real-world computing problems while playing an active role within a professional software developer community. FOSS projects are characterized by being community-based, open to new contributors, offering a high degree of transparency given their open-source nature, being globally distributed, and encouraging student initiative through a meritocratic process. However, FOSS practitioners report that despite many highly-attractive features of such projects, there is a significant number of barriers to a broader adoption of FOSS in computing education. Here, we describe a number of techniques that help address some of these barriers.

## 2 SOFTWARE ENGINEERING STUDIO

Software Engineering Studio at Central Connecticut State University is a framework that connects external clients with teams of undergraduate students working on software development projects. Since 2014, over 90 teams consisting of 4-5 students worked on nearly 60 unique projects for 35 distinct clients. 24 of these clients are non-profit or community organizations; most of their projects were developed using the FOSS model. Student teams work closely with their customers to create working solutions from inception through the software development life cycle to deployment and ongoing maintenance. Over the years, these externally-sourced projects evolved into a highly-scaffolded framework that provides enough structure to ensure project success while being flexible to provide teams with enough agility for adapting to evolving requirements and addressing any emerging challenges.

Students from two senior courses, Software Engineering (SE) and Senior Project (SP), work on all projects supported by the Studio. Student teams in the SE course are formed during the first week. The rest of the 15-week SE course consists of a series of 2-week iterations as shown in Table 1. The SP course consists of seven 2-week development sprints. A typical project lasts two to three semesters. After the first semester, students who worked on a given project in the SE course, continue into the SP course and work with new SE student teams ensuring project continuity and knowledge transfer. Ongoing maintenance issues are typically addressed by teams in the SP course. We believe that our scaffolded agile approach addresses a number of FOSS challenges described below [1–3].

### 2.1 Steep Learning Curve

One of the main challenges of FOSS projects is the steep learning curve concerning a diverse range of project characteristics. For example, the project domain complexity, significant scale of the project codebase, and diversity of technical tools may be overwhelming. In our approach, we gently nudge student teams to incrementally dive deeper into the project and understand its requirements through frequent and direct interactions with the project client. The same sequence is used to help a new team gain a high-level understanding of the existing codebase and the corresponding requirements along with the new or changing features that they will be responsible for. Three deliverables (numbered 2-4 in Table 1) are aimed to provide a reasonable on-ramping to prepare teams for more technical aspects of the project.

### 2.2 Project/Course Synchronization

A community-driven structure of a FOSS project may not always be a good fit for the relatively rigid schedule of a typical course. Because FOSS project communities are comprised of volunteers, there is never a guaranteed time frame when student questions will be answered or how soon student coding contributions can be approved by the project maintainers. By providing a development sprint structure to the project and by ensuring that the core of project contributors is comprised of students, it becomes easier to


*ITiCSE 2022, July 8–13, 2022, Dublin, Ireland*

ACM ISBN 978-1-4503-9200-6/22/07.
https://doi.org/10.1145/3502717.3532163

| Week | Project Element | Scaffolding/Deliverable |
|---|---|---|
| 1 | **Introduction** | Students teams are formed |
| 2-3 | **Inception** | 1. Student team agreement |
| 4-5 | **Introduction to agile** | 2. High-level requirements overview |
| 6-7 | **Requirements outline** | 3. Product backlog |
| | | 4. Refined product backlog |
| | | 5. Mid-point presentation |
| 8-15 | **Development (4 sprints)** | 6. Sprint plan/backlog (x4) |
| | | 7. Sprint review (x4) |
| | | 8. Sprint retrospective (x4) |
| | | 9. User and developer documentation |

**Table 1: Scaffolded structure of a software engineering course project**

provide a reasonable cadence to all project participants and align it with a typical course schedule.

## 2.3 Evaluation and Grading

Computing education practitioners using FOSS projects in classroom report that it is often challenging to properly evaluate student contribution to the project. What if their code modifications were rejected by the maintainers? What if the feature that they were working on for several weeks is no longer needed? In our approach, we require students to produce a number of deliverables, both as written reports and as presentations. These deliverables provide a rich selection for various aspects of grading that allows the instructor to evaluate students' technical writing, design skills, communication skills, not to mention their technical contributions to the project codebase.

## 2.4 Project Scope Management

While the majority of HFOSS projects focus on large-scale projects with a multitude of stakeholders, our efforts focused on working with small non-profit and community organizations. In addition to increasing student motivation due to a significant community engagement factor, our approach is advantageous from several perspectives. Students work directly with a clearly-identified project client who serves as a one-stop shop for addressing most concerns regarding the project functional and non-functional requirements. Most importantly, a direct contact with the client provides instructor with an opportunity to negotiate or adjust the project scope early enough to make sure that it meets both customer expectations and student capabilities. Our approach provides a scaffolding of various deliverables scattered throughout the project that requires students to constantly communicate with the project client.

## 2.5 Reasonable Instructor Involvement

Many approaches to using FOSS in education require instructor to negotiate with the existing project community about the kinds of contributions students can make, the level of support, and the frequency of communication that the community can provide. Some FOSS practitioners advocate for the instructor to be responsible for formulating all project requirements to ensure that student teams stay on track in a typical community-driven FOSS project. These factors may result in a very significant commitment on the instructor's time and effort. Our scaffolded approach helps alleviate this issue. By controlling the project scope from the outset, the instructor can ensure that the overall set of requirements is feasible, which will be reflected in the refined product backlog. Given a reasonable scope of the project backlog and taking into account priorities and product values identified by the project client, student teams identify specific goals and work toward achieving them during several development sprints. At the end of each sprint, teams present their outcomes to the client during a sprint review and reflect on the lessons learned during a sprint retrospective. These elements are an integral part of our scaffolded approach ensuring that teams stay on track and make a measurable progress. This also helps ensure that the instructor's involvement in the project management is reasonable because s/he can rely on each team to formulate detailed project requirements and commit to each sprint goal.

## 2.6 Project Outcomes

Despite the many advantages of FOSS projects from the student learning perspective, the amount of students' tangible contributions to the end goals of a typical project is usually limited. This is due to the overall project complexity, lack of a clear project timeline, and the open community-driven nature of the project. Our approach helped us achieve successful outcomes for a great number of projects where we worked with local clients who were committed to being engaged with student teams throughout the duration of each project. Project scaffolding helped ensure that for every project increment encompassing a given semester, all stakeholders were on the same page regarding the scope and the nature of the project features and about the project progress throughout every sprint.

## REFERENCES

[1] 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 (jul 2018), 31 pages. https://doi.org/10.1145/3145476

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

[3] Lori Postner, Darci Burdge, Heidi J. C. Ellis, Stoney Jackson, and Gregory W. Hislop. 2019. Impact of HFOSS on Education on Instructors. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE '19)*. Association for Computing Machinery, New York, NY, USA, 285–291. https://doi.org/10.1145/3304221.3319765