**International Journal of Information and Communication Technology Education**
Volume 16 • Issue 2 • April-June 2020

# Open Sourcing the Pedagogy to Activate the Learning Process

Alan Rea, Haworth College of Business, Western Michigan University, Kalamazoo, USA
Nick Yeates, UMBC, Baltimore, USA

## ABSTRACT

Information systems graduates increasingly need to understand the collaborative, technology-driven practices inspired by open source software development that are fundamentally changing today's workplace. To meet this challenge, instructors must bring open source principles and technologies to active learning experiences. In this paper, the authors describe how nineteen undergraduates in a web development and design course at a Midwest university worked collaboratively with leading open-source software provider, Red Hat, to revamp the Teaching Open Source website. Accommodating this semester-length project required making significant revisions to course structure, instructional strategy, and assessments. The authors also describe the challenges of integrating these practices into the classroom and conclude with project reflections, including cautions and suggestions for instructors considering similar initiatives to move away from the "instructor as expert" paradigm to "meritocracy rule" thereby enabling students to make decisions with impacts beyond the classroom.

## KEYWORDS

Active Learning, Agile Development, Classroom Project, Collaboration, Inner Source, Open Source, Pedagogy, Web Development

## INTRODUCTION

The information systems department offers an undergraduate web design and development class at least once a year. The course is a degree requirement for juniors or seniors majoring in Digital Marketing and an elective for Information Systems majors. Instruction focuses on client-side scripting such as HTML5, CSS3 and JavaScript, and web design theory.

To practice what they are learning, students typically work in small groups on a semester-length project to develop websites for local small businesses or nonprofit organizations. However, at the outset of a fall semester, an opportunity presented itself for students to collaborate on a more substantial project: building a website for the only billion-dollar open source company, Red Hat (redhat.com).

This paper describes how an instructor and Red Hat consultant developed a course structure that enabled nineteen undergraduates to use open source software (OSS) development principles and technologies as they redesigned, developed, and implemented the Teaching Open Source website (teachingopensource.org) according to Red Hat's specifications and input. Students worked within an agile development environment much like they would in the real world (Turnu, et al., 2006) and were encouraged to make their own decisions to meet project expectations.

DOI: 10.4018/IJICTE.2020040101

In the rest of Section 2, the authors describe the project's context and learning strategies. Section 3 contrasts it with previous attempts at collaboration. Section 4 explains the division of labor (into five student groups) and the means of motivating them. Section 5 describe new processes put in place to ensure the students succeeded at collaboration. Section 6 summarizes how technologies allowing virtual collaboration were applied. Section 7 reviews overall findings, including successes and shortcomings, and Section 8 offers suggestions for replicating the learning experience. Finally, Section 9 offers conclusions and future directions.

## Open Source and Inner Source Approaches

To examine how the concepts and practices that the authors used diverged from established class projects and pedagogies, this subsection summarizes the industry strategies and practices that the class employed.

Open source typically refers to how source code is distributed and shared publicly via open source licensing (Koohang & Harman, 2005; Kamthan, 2007; OSI, 2019). Similarly, a new business practice--the "open source way"--applies to the work environment the same methodologies, best practices, processes, tools and culture that have transformed software development. The open source way empowers a workforce to collaborate freely as a community of people; new priorities include transparency, communal work, meritocratic rewards, and rapid prototyping (Red Hat, 2009).

These methods also may be used when the code will not be released to the public, or at least not to more than a select few (InnerSource Commons, 2019). The InnerSource approach describes communal, transparent, and iterative methods applied to in-house platform development (O'Reilly, 2000; Stol & Fitzgerald, 2015). Inner source provides the benefits of open source for circumstances in which company culture, technical reasons, legal uncertainties, or business secrets prevent disclosing the source code outside the company (Capraro & Riehle, 2016). Another instance is when a government has restricted the material from open source licenses.

We classify the project results as inner source because students worked with the authors in an online meeting space accessible only to themselves and project facilitators from Red Hat and the Professors Open Source Software Experience group (POSSE). The website was inner source because its code has not yet been released, but open source because it will be freely available after its completion and approval by Redhat and POSSE.

## Project Origins

The authors met at a POSSE (foss2serve.org) workshop for computer science and computer information systems professors sponsored by Red Hat and run by POSSE member faculty with support from the National Science Foundation. Without participating in initiatives designed to foster partnerships between academia and industry, projects such as this one do not easily occur. The authors encourage instructors who want to implement classroom projects such as this one to attend meetings of local AITP chapters (aitp.org) or contact organizations looking for project assistance, such as the Free Software Foundation (fsf.org)

During the workshop Red Hat and POSSE spoke of an initiative to revamp the Teaching Open Source website. One of the authors (the instructor) saw an opportunity to engage his students in the redesign and offered to take on the project in the fall. After discussions with Red Hat managers and POSSE leadership, the instructor and Red Hat consultant left the workshop as project leaders and agreed that the collaboration would be:

- Implemented by students in a classroom project environment;
- Managed by Red Hat via the consultant and the university via the instructor; and
- Overseen by stakeholders and facilitators from Red Hat and POSSE.

### Project Approach

The project would be considerably larger than the instructor's previous web projects, which had small teams of three to five students using limited deliverables to design websites for local clients. For this more substantial undertaking, the authors agreed that the class should function as an open source community in which diverse people collaborate for an improved outcome (Sack, et al., 2006).

Revisions to the course structure, instructional strategy, schedule and assessments would be necessary. The authors first planned the project's approach and expectations, which would take their cues from the IT industry's complex and multifaceted processes both in OSS development (Stewart & Gosain, 2006) and recently, InnerSource development (Capraro & Riehle, 2016; Stol, Babar, Avgeriou, & Fitzgerald, 2011). The authors soon determined:

- Everyone would work on the same project;
- Everyone would get the same project grade;
- Five work groups would have three to five students each;
- Each group would have a subfocus;
- Groups would work with Red Hat to implement the website;
- Email responses would be expected within three days;
- Weekly group sprints would be a check-in with everyone;
- The consultant would make an in-person visit at kick-off.

### Project Expectations

The student team was asked to begin the project by investigating the existing site, evaluating its content and writing a site analysis. The student team's first deliverable report states:

*The original Teaching Open Source website was put together to encourage students to learn about open source projects and software by reaching them through teachers. The site was mainly a teaching resource to allow teachers to view projects, recent open source events, and blogs regarding open source. Changes in management of the site led to lowered upkeep, whereupon spam accounts and poorly structured data led to most users leaving the site...[The client] looks to relaunch the website to better facilitate the original concept of promoting education and problem solving with open source...*

After summarizing the tasks at hand, the analysis delineated objectives for each student group (as discussed in Section 4.1) as well as for the overall team. It also offered a preliminary timeline to meet the assigned delivery date. After feedback from the authors, students discussed the report at length and made revisions. By allowing students to set expectations and work through them with the instructor and consultant, the team had a sense of ownership from initial stages (Stewart & Gosain, 2006).

## PREVIOUS COLLABORATION APPROACHES

Case studies have focused on team building among students pursing different college majors and with programming levels ranging from beginner to advanced, as were the challenges with the students. Sahin (2011) recommends taking students' preferences and instructor's considerations into account when forming teams for software engineering courses. Kruck & Teer (2009) advocate the use of interdisciplinary teams as an advantage in a group technical project and describe how a professor can adjust a course to accommodate it. Fortunately, the authors were able to incorporate this knowledge using the CATME system (CATME, 2019; Layton, Loughry, Ohland, & Ricco, 2010) for generating a questionnaire that classified each student's skills and knowledge as well as attitudes toward team projects (catme.org).

Team building is only the first hurdle in many classroom projects; despite being part of a traditional in-person class, the project required online collaboration. In seeking to replicate an efficient interactive learning environment as outlined by Uzunboylu (Uzunboylu, Bicen, & Cavus, 2011), the authors' experience was that Google Docs, Hangouts, and Sheets, as well as LucidChart and Trello were invaluable. The Google offerings, in particular, made the project possible (as explained in Section 6).

A final goal of student teamwork in a virtual environment is to motivate students to become active rather than passive learners (Pundak, Herscovitz, Shacham, & Wiser-Biton, 2009; Schiller, 2009; Coldwell, Craig, & Goold, 2011; Drake, 2012). While sharing goals with the team during the semester, the authors encouraged students to determine how they would reach them, in particular by deciding which tools and components best fulfilled the client's requirements (Teel, Schweitzer, & Fulton, 2012; Kamthan, 2007; McComb, Green, & Compton, 1999). This was validated when students disputed the client's suggestions. Some, but not all, of their arguments were adopted. During post-project feedback, one student noted that the authors hearing and giving serious consideration to their ideas and suggestions during sprint meetings (explained Section 5.3) was positive. Such support is critical to OSS development teams because it "fosters trust and good communication practices by encouraging behaviors and orientations that are beneficial to the team's work" (Stewart & Gosain, 2006, p. 292).

## Previous Classroom Attempts

Semester-length, real-world projects are core to the web design and development class and identified as such in the course description. However, preceding projects were smaller scoped and less complicated; a few students at a time had been able to manage the information architecture, design, development and implementation of a local organization's website assigned to their team.

Yet, previous projects also had overly depended on self-generated initiative from both clients and students. Some clients had been generous--and others uncharitable--in volunteering time and assistance to project teams. Moreover, past projects lacked iterative deliverables or checkpoints to help teams stay on schedule and on task. Ultimately, these learning experiences lacked community, cooperation, and a strong sense of identification with the project team: three concepts essential for OSS development to be effective (Stewart & Gosain, 2006).

Taking this into account, the authors improved the class structure and assessments through smaller deliverables, planned revisions of deliverables, weekly check-ins with the client and flexible objectives. Many of these techniques mirror tenets of agile development in software engineering (Kamthan, 2007; Teel, et al., 2012). Most worked well, but Section 7.3 of this paper describes times when the project deviated from its goals.

## Industry's Shifting Paradigm

The course's new strategy borrowed from the Open Source, InnerSource, Agile, and DevOps approaches: all new models for software development as modern IT projects get bigger and more complicated. Industry use of organizational and cultural practices taken from open source originated with the software itself in the late 1990s and became a trend along with the InnerSource model in the following decade (Capraro & Reihle, 2016; InnerSourcing, 2019). Written in 2001, The Agile Manifesto (Beck, et al., 2001) created a new trend in project management that has benefited software development-based organizations by promoting iterative, cooperative, and frequent feedback practices.

More recently--beginning in 2009 in Belgium (Edwards, 2012)--the rapidly-growing DevOps movement has helped software organizations resolve the conflict between product development (feature creation) and operations (systems administration) that often delays getting software to the marketplace. DevOps principles are being embraced by large, complex IT organizations worldwide.

Today, large industry players such as IBM, Intel, Facebook, and Microsoft are more committed than ever to open source software. In recent years they have begun to allow their own developers to share high-potential code via company-sponsored initiatives, such as IBM's developerWorks Open, or

by joining existing open source software communities (Capraro & Riehle, 2016). Preparing students to work in these new fast-paced approaches to software and system development is important for their continued success. However, it does require instructors to shift the overall classroom focus and become more of a guide rather than the sole expert. The next section outlines how the semester project created an environment where this could happen.

## PROJECT FRAMEWORK

The semester began with forming student groups, setting up project architecture, and familiarizing students with the context of the project, as detailed in the following subsections.

### Group Focus Within the Team

While intending to encourage the entire class to collaborate on activities such as brainstorming and problem solving, the authors also saw value in forming smaller groups to accomplish specific tasks. Therefore, by applying an accepted approach to OSS development that uses loosely prescribed roles (Sack, et al., 2006), the authors devised the following groups of three to five students each:

- **Information Architecture / User Experience (IA):** After first harvesting material from the old site, the IA group focused on content organization and usability, resulting in a new information structure and improved site navigation. Its members created an impressive 1,188-cell spreadsheet mapping old content to new, which helped the entire team rearrange the site;
- **Project Management (PM):** This group managed schedules and deliverable dates and ensured collaboration across groups. One of its members attended each of five sprint meetings per week, during which they logged progress and assisted the team with reminders and next steps;
- **System Administration, Programming, Development (SysAdmin):** This group was responsible for cloud infrastructure implementation, shell-level administration tasks, and verifying that the site was secure, robust and reliable. The group installed and tested all platforms, modules and plugins, and managed user roles. Moreover, the group independently learned OpenShift (openshift.com), an open source cloud-based containerized application hosting system, and implemented and configured Wordpress (wordpress.org), the content management system (CMS) platform selected for the new site;
- **Systems Analysis (SysAnalysis):** This group researched and tested seven platforms with the potential to meet client requirements and worked with the authors to create a 1,640-cell spreadsheet analyzing these findings. After recommending a CMS, the group documented and explained the system to both potential system administrators and users. This accomplishment should not be minimized given the lack of clear documentation in many open source projects (Izquierdo-Cortazar, González-Barahona, Robles, Deprez, & Auvray, 2012);
- **User Interface and Design (UI):** This group was responsible for design, layout and color theme, as well as the site achieving client standards for usability and branding. Dozens of designs were sketched on paper and then built in LucidChart (lucidchart.com), a collaborative online diagram maker similar to Microsoft Visio. Additionally, group members independently learned Wordpress's theme and plugin framework to take their art from idea to implementation.

Although forming groups helped to break a complex project into more manageable pieces, specialized groups also risked creating a silo effect within the team. The authors avoided this by requiring collaboration on all documents and group reports at weekly sprints (discussed in Section 5.3). For example, the class received proposed designs from the UI group, organizational revisions from the IA group, and summaries of sprints from the PM group. All of these items were presented

**Figure 1. Old to new information architecture mapping**

| Hierarchy | New Page Name | Old Page Name(s) |
|---|---|---|
| | | |
| Teaching Materials | Teaching Guides | currently on github, Teaching Materials Catalogue |
| | Teaching Materials Catalogue | Teaching Materials Catalogue |
| | In-Class Learning Activities | currently on foss2serve |
| | Academic Papers List | Articles, OSS education papers in scholarly journals |
| | Textbook | POSSE Textbook |
| | List of Courses | Course Content |
| | Schools That Teach OS | Programs That Teach OS |
| | Intro to FOSS | |
| | | |
| | | |
| | | |
| | Promote POSSE / Foss2Serve | Programs About Teaching OS |
| | Presentations | Presentations |
| | | |
| Participate | Join IRC Chat | Join TOS IRC Channel |
| | Participate in Initiative | Participate in Project |
| | Presentations | Presentations |
| | Promote POSSE / Foss2Serve | Programs About Teaching OS |
| | | |
| | Edit this site | Join Wiki |
| | Events | Upcoming Events |
| | Blog Feed | Planet, Blog onto Planet / Planet Feed List, How to n |
| | Join Mailing List | Mailing List Archives |
| | | |
| Connect with Others | Roll Call | Roll Call |
| | Join Mailing List | Mailing List Archives |
| | Blog Feed | Planet, Blog onto Planet / Planet Feed List, How to n |
| | Events | Upcoming Events |
| | | |
| | | |
| | | |
| | Promote POSSE / Foss2Serve | Programs About Teaching OS |
| | | |
| Friendly Projects | OSS Project List | FOSS Mentor Projects |
| | Dating Site Stub... coming in 2016 | |
| | | |
| | Promote POSSE / Foss2Serve | Programs About Teaching OS |

to the class as a whole and could be viewed at any time in the shared project folders. Peer cooperation throughout the semester was a critical component to ensure collaborative work (Poindexter, 2003).

## Collaboration Motivators

If the student groups failed to work in a joint-delivery environment, and siloed outputs had to be assembled at the project's end, there likely would be compatibility issues (Stol, Avgeriou, Babar, Lucas, & Fitzgerald, 2014). Therefore, each student had to understand collaboration benefits. Although there were many motivational factors inherent in the course structure, the three most important were:

- Shared project grade;
- Shared document space; and
- Shared project knowledge.

Ideally, students were motivated by interests that *they* cared about, and not simply told to collaborate. Still the authors found these three shared areas to be the most important when facilitating open source interdependence throughout the course.

**Figure 2. Content management system analysis**

| Major Requirements | | | | |
|---|---|---|---|---|
| **Features / Abilities** | **Description** | **Joomla** | **Wordpress** | **Silverstripe** |
| **CMS Overview** | | | | |
| FOSS/ License | Free/Open Source? What License does the software operate on? | Pass: GPL | Pass: GPL | Pass: BSD License |
| Developer/Community Strength | Developer commits often and recent? What is the community support like? [1] | Pass+ Timeline for release into 2016 | Pass+ very popular with routine commits | Pass: Commits often, different versions available |
| OpenShift Hosting capability | OpenShift Immediate Compatibility | Pass [2] | Pass | Depends [3] |
| Platform | Platform (java, php) | php | php | php |
| **Subjective usability of CMS by admins and editors** | | | | |
| Content Editor | Content editor usability-preferrably low barrier to entry [4] | Pass | Pass+ | Pass |
| Developer | Developer usability [5] | Pass | Pass | Depends |
| **Publishing & Content Management** | | | | |
| Wiki-like Editing UI | WYSIWYG (What You See is What You Get) On/Off [6] | Depends | Pass++ | Pass- [7] |
| Customizable to our CSS | Can we use custom CSS, or use a theme [8] | Pass | Pass | Pass |
| Images in Content | Ability to insert images to content | Pass | Pass | Pass |
| Define Access/Permissions Based on Roles | Control privilege to content by defining roles, access, or editing rights [9] | Pass | Pass | Pass |
| Restrict Content editing by Type/Directory Structure | Control content access by content type/directory structure, such as static pages like Contact Us [10] | Pass [11] | pass | Pass |
| Broken Links | Detect and Notify broken links across site [12] | Pass | Pass [13] | Pass [14] |
| Mobile View and Nav | Viewable and Navigable on Mobile platforms | Pass | Pass | Pass [15] |
| Page/Title rename or move | Page or Title rename/move [16] | Pass | Depends-[17] | Pass [18] |
| **Document Mgmt** | | | | |
| User-Defined Metadata | Metadata types can be user defined | Pass | Pass | Pass |
| **Archiving & Auditing** | | | | |
| Track Site Changes | Track Changes to documents, version types | Pass | Pass [19] | Pass [20] |

## Shared Grade

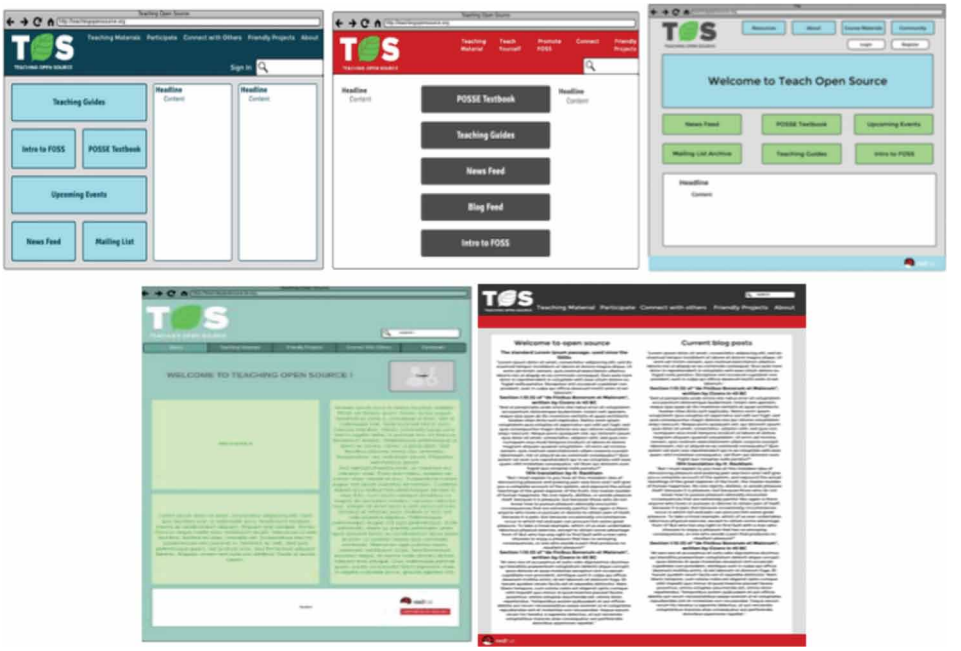
Arguably the single most important motivator was the shared project grade. The class earned a single mark for the all of the project deliverables, which accounted for 50% of each student's final grade for the course. Early in the process, the authors considered whether the instructor should grade separately on certain deliverables from each group (e.g., site hierarchy analysis by the IA group), but rejected this to encourage cooperation and avoid a silo effect. It worked well. Asked about the benefits of participating in the project, a student noted:

*It was, to be blunt, scary just letting go of all control over more than half of my points [for the course] on this project by allowing others to work towards it instead, but it was a great learning experience.*

## Shared Document Space

"Default to open," or transparency in decision-making, is a principle that Red Hat applies to its entire work culture. Students were strongly encouraged to work only in shared spaces. The authors told them, "If it is not written down for all to see, it didn't happen." If during sprint meetings a student disclosed working apart from the group, the authors asked him or her to post the work--even as a rough draft--in a shared space. This occurred only once or twice before all work was shared by default.

**Figure 3. Proposed site designs**




To enhance teamwork further, edit privileges were granted across the collaborative cloud platform. No one misused or abused these privileges, an issue of common concern in open source processes. Shared document space allowed students to follow the overall progress and find opportunities to contribute to content not assigned to their group (Poindexter, 2003). This permitted meritocracy, a unique benefit of open source communities. At any point in time a student was able to provide feedback on any area of the project. This leads to process openness.

## Shared Knowledge

Transparency is another much-touted advantage to open source communities; it helps members multiply their personal skills to achieve a greater result than if they had worked separately (Stol, et al., 2011, 2014). Creating a culture of openness in the Red Hat project transcended students simply knowing what others were doing or being able to monitor real-time progress on deliverables and provide input. Students had to understand that helping each other *was* in their own personal interest because a better project would be the result. As the project progressed, the authors saw an increase in peers giving feedback on other group deliverables either via the shared space or in class sessions.

## Project Processes

However, an increase in peer cooperation would not have happened without an adequate atmosphere to encourage collaboration (Poindexter, 2003). Without careful planning and facilitation, students will become frustrated and ultimately stymie the learning experience for all. The following subsections describe how the project was allocated adequate weekly time and resources to enable peer cooperation and increased collaboration beyond what students are accustomed in a more traditional classroom environment. Although many smaller course adjustments are always necessary, the authors assert that without a shift in course scope and weekly sprints, this project would not have been completed.

### *Class Scope*

Collaboration typically requires more "overhead" efforts, such as planning and meetings, which do not directly result in a student's measurable progress on a project deliverable (Pundak, et al. 2009). As such, the project with Red Hat required significant changes to the instructor's strategy for the course, including grading procedures.

Modifying the class schedule to allow more classroom time for the project was slightly easier than deciding how much it should influence final grades. In previous semesters, a smaller group project counted for 35% of a student's final grade with the remaining 65% comprised of exercises, hands-on labs, and exams. After a discussion with students that ended in a show of hands, the Red Hat project was adjusted to become 50% of the grading component for the course. The instructor accomplished this change by removing exams from the course. Ultimately this change enabled the authors to increase the number of iterative sub-deliverables for the project in order to commit early, commit often (Red Hat, 2009), as well as allocate more time to enable group peer cooperation and team collaboration on the project in class.

### *Client Visit is Critical*

Because the project was to be a major class focus, a client-team meeting early in the process was essential. Fortunately, the consultant was able to travel on short notice and became an "executive in residence" for three days at the university. Besides meeting with various student groups, he led in-class workshops to discuss open source and Red Hat. In the Web class the authors conducted workshops to help students understand the project though question and answer sessions, set benchmarks, and help students begin to brainstorm about the project (Kamthan, 2007).

This face-to-face meeting fostered trust between the team and the consultant that would boost students' motivation to collaborate throughout the semester. As Stewart & Gosain (2006) noted, "Affective trust stems from emotional attachment between a trustor and a trust target and may, therefore, be most relevant to potential developers' psychological and emotional reasons for joining, staying with, and contributing to OSS teams" (p. 296).

### *Sprint Meetings Are Anchors*

To maintain the close interaction, a weekly sprint conducted in a round robin format was scheduled with each group and the consultant. During the sprint, group members each shared what they worked on that week and what they anticipated doing the next. The consultant led the initial group sprint, but after the first meeting asked each group leader to take it over, thus maintaining the culture and pace of the weekly sprint.

During the weekly sprint, in addition to the consultant, a PM group member also was present. This requirement was a highly effective means to record and disseminate information among the groups and the team as a whole. The PM group kept a detailed log of all meetings, resulting in a virtual project history that would benefit anyone wanting to learn from reviewing the process (Teel, et al., 2012). Open source culture advocates such transparency because it allows new developers, transient bug fixers, curiosity seekers, and researchers alike to delve deep into a project knowledge base and culture as if they were participants from the onset. Thus, OSS can expand beyond its progenitors (Stallman, 1992).

Asked at the end of the project to share what worked well, nearly every student named the weekly sprints:

*The thing that worked well the most with this project were the weekly sprint meetings…[The sprints] allowed us to get feedback on our work very quickly which in turn allowed us to make decisions on what to do next. I have taken part in many large projects where a deliverable is given and there is very little to zero communication up until the day before a deliverable is due. This causes everyone to*

*scramble at the very last second, which usually results in low quality work. I also believe the workshop days where the entire class was able to collaborate and speak to [the consultant] were equally as important. I also believe this more accurately represents a scenario you would face in the real world.*

The consultant's availability to each student group via its weekly sprint required him to attend five Google Hangouts per week, an exceptional commitment that contributed greatly to the project's success. Although the authors realize a large commitment such as this may not be possible for every class project, it was necessary for one of this scope. However, sprint meetings could be scaled back depending on the client's goals for the project.

## COLLABORATIVE TECHNOLOGIES

Project commitment and open lines of communication (Stewart & Gosain, 2006) are tenets of successful open source projects. Collaborative technologies enabled the students to accomplish the latter despite most of their communication and work taking place apart from their twice weekly classes.

Cross-platform compatibility enabled students to interact using any operating system or modality, including Chrome OS and smartphones. The class defaulted to productivity tools offered by Google Apps for Education (google.com/edu) because of its availability and Google's strong communal-oriented creation abilities. Google enabled the students to collaborate on the same documents to create, share, and edit files in real-time; however, any other multifaceted development environment would work.

Another preference was open source platforms such as OpenShift, Git, and Wordpress. The following table summarizes how technologies allowing virtual collaboration were applied to the project. Each is rated (High, Medium, Low) in terms of importance.

Collaborative technologies allowed students to witness project elements evolve and helped them to contribute in real time rather than waiting for a deadline. These communal social motivators are critical to the success of open source and other collaborative projects (Krogh, Haefliger, Spaeth, & Wallin, 2012).

## OVERALL PROJECT FINDINGS

Given the fluid nature of open source processes in play not only in the project components and interaction but also the pedagogical shifts to encourage them, we offer both the major successes and the shortcomings for those who might want to implement similar initiatives. In this section, the authors distill the primary motivators and potential sticking points in regard to replicating similar learning opportunities.

### Instructional Shift

Collaborative and cooperative learning requires not only students but also instructors to take on new roles. Early in the semester the instructor abdicated power as content expert of the classroom in favor of becoming the designer or architect of a learning experience (Poindexter, 2003). The quality of the collaborative process became the instructor's priority as he helped students make decisions (Lakhani & von Hippel, 2003). At times this included letting the students fail, learn how to recover, and return to the project's critical path.

Even if the instructor's hands-off instructional approach resulted in a less-than-perfect final project, students benefitted from having observed the ramifications of poor decision-making in a real-world OSS development process. Krogh, et al. (2012) note that the potential impact of the work is a strong motivator. As one student observed:

**Table1. Collaborative technology used**

| Tool | Explanation | Importance |
|---|---|---|
| Google Docs | All work was to be placed in shared folders with edit/revision privileges extended to all. This included the contents of group subfolders and even a folder for turning in specific deliverables. Because Google Docs logs edits with attribution, students could see who made any changes, including deletions. To address potential conflicts, the Project Management (PM) group wrote policies for handling revisions, code commits, and other project needs; this group also was responsible for quality control on all final submissions. | High |
| Google Hangouts | Weekly sprint meetings were held as group video chats on Google Hangouts, giving students with busy schedules the flexibility of attending via smartphones. For workshops throughout the semester, Google Hangouts allowed the consultant to address the class from a projection system, and later to move around the room on a laptop screen to meet with groups. | High |
| Google Calendar | Students could check deadlines and conveniently join Hangouts via links from Google Calendar. PM members and all group leaders had edit privileges to share dates and times. | Medium |
| Trello (trello.com) | Early in the semester a Trello board was useful to organize deliverables, group formation, and other information. Its use diminished as the project progressed. | Low |
| LucidChart (lucidchart.com) | The UI group and IA group extensively used LucidChart for collaboration on site schematics, theme building and CSS frameworks. An educational version made LucidChart available to students without cost to them or the university. | High (Select Groups [IA and UI])<br>Low (Remaining Groups) |
| Git and GitHub (github.com) | All code was hosted in a private Git repository; students used GitHub to share code, documents and other materials. Using Git and GitHub was challenging for students and required in-class instruction using tutorials and practice exercises. Concepts such as fork, push, and pull denote actions that are confusing for first time users. GitHub recognizes the challenge many face and has developed detailed guides and tutorials at its website (education.github.com). | High (Select Groups [SysAdmin]) |
| OpenShift (openshift.com) | The project's development and production sites were housed by OpenShift, which is Red Hat's open source cloud-based Platform as a Service (PaaS) hosting system. It provides quick access to a Linux shell prompt and can automatically set up development environments such as PHP and a MySQL or PostgreSQL database. Other free PaaS offerings such as Heroku (heroku.com) are also available. The SysAdmin group, in particular, used OpenShift to test multiple Content Management Systems (CMS), and to stand up a development and production copy of Wordpress for the project. OpenShift and other PaaSs let students access as many virtual servers as needed to experiment with and deploy project technologies. | High |
| Email | All groups and the team as a whole relied on email sometimes to its detriment (Section 7.3). | High |

*The most important experience of this project was the real-world work environment we participated in. There are not nearly enough courses that truly apply a real-world environment to the curriculum and this class was an exception. This project allowed us to proactively learn what working in a web development position would be like. … Overall this gave us valuable career applicable knowledge we can take with us to a professional environment.*

The consultant first expected to function as "client" for the project but soon took a more active project manager role that added another level of real-world authenticity. In addition to being an open source business strategist at Red Hat, he also had previously worked on an InnerSource project at the Naval Air Systems Command (NAVAIR) of the U.S. Department of Defense. He interacted with students in a truly open source collaborative manner in which the merit of ideas and approaches influenced decisions (Raymond, 2001). For example, the UI group used research and examples to convince the consultant that the client should go with a different site design than the preferred one. His willingness to allow ideas to flourish contributed to student ownership and the project's ultimate success.

## Project Successes

From student, industry, and instructor viewpoints, the project worked well. The authors note as examples:

- **Student ownership and excitement:** Encouraged to make team decisions, listened to as valued team members, and trusted to proceed on their own in groups and in a team (Stewart & Gosain, 2006), students responded by making it their mission to complete a successful project;
- **Real world lessons and output:** The project extended the relevance of doing web design and development well beyond the classroom, allowing students to learn course content by *doing* rather than by only reading and practicing. Although coursework besides the project included labs, the potential for real-world results made the OSS learning experience indelible because it enabled students to try diverse approaches as they engineered the product (Kamthan, 2007);
- **More employable graduates:** Completing the project gave students an industry-sponsored project to put on their resumes and to help build their professional identities on LinkedIn (linkedin.com). Long notes that OSS experience on student resumes has a major impact when students are looking for their first job (2009). Additionally, the consultant was asked to submit recommendations that led to students getting technology jobs and internships. As one student shared:

*I lack internship experience and most of my job experience is in entry level service industry positions. Having been part of this project gives a great experience to list on my resume, something I consider highly important as I write this paper not a week removed from taking my walk at commencement.*

- **Benefits to client:** Red Hat and POSSE received an improved product created by a team of nineteen aspiring professionals, any of whom could be recruited as interns or future employees. Red Hat also was able to encourage good software practices, a benefit to the entire industry (Long, 2009);
- **Benefits to university:** Well beyond a press opportunity, the project further helped to foster a culture of collaboration between the college and its industry technical partners beyond those in the local community. Additional university entrepreneurial ventures are now considering open source collaborations as potential venues;
- **Proof of open source success:** The authors offer this project as an example that open source, coupled with collaborative and peer learning, can be successful. Though a significant departure from traditional instruction-based classrooms, projects such as this one should be more commonplace on campuses wanting to prepare students for today's workplace (Long, 2009). The

project accomplished this while also applying the principles of OSS software development, as the best enterprise development organizations also do (Capraro & Riehle, 2016). The authors' hope is that other instructors, familiar with this project's success, consider the viability of open source in the classroom not only with technology but also with pedagogy.

## Project Shortcomings

Although successes outweighed shortcomings, a few problems did impact student collaboration and project deliverables, as described below:

- **Feature bloat and scope creep:** In the IT industry, project management suffers when the product accrues too many features or changes uncontrollably. In the students' project, the team wasted time studying features such as user management that were outside of the project scope and thus counterproductive. In hindsight, the authors should have moved more quickly to define the project scope rather than permit the team to promote features that were not required;
- **Critical path slowdown:** In feedback at the project's conclusion, nearly every student said it had taken too long at the project's outset to decide which CMS to use:

*...what we could have done better was the time comparison between researching the different parts, especially the CMS, and actually developing the prototype...too much time was spent on it, leaving the [team] very little time to create and implement the actual prototype.*

At the time, the authors were not aware of the slowdown and encouraged some of the research. Care should be taken to start work into developing technical aspects early on, versus spending too much time weighing which paths to go down.

- **Group arguments:** Although minimal, there were instances of group conflict. For example, the SysAdmin group argued for a certain CMS that the SysAnalyst group did not value from its research. The authors let the students work through their conflicts without interference in order to maintain open source governance structures (O'Mahoney & Ferraro, 2007);
- **Email miscommunication:** Students relied on Google docs to stay abreast of collaboration on the deliverables, perhaps to the neglect of email. Discussion lists were not created for each student group or for the overall class. As a result, forgetting an email address in the *Cc:* field would leave a student uniformed. Discussion lists also would have contained a record of communications similar to the virtual project history described in Section 5.3. Future replications must have robust communication systems such as email lists or newer technologies such as Slack (slack.com);
- **Privacy rights / FERPA:** Getting students' consent to use personally identifiable information for educational or promotional purposes should not be overlooked. Consider checking with university counsel in terms of what FERPA items can be released or waived by the students. If survey instruments will be used, contact the university's human subjects research board.

The overall project was successful because students learned OSS processes and techniques, and as well how to collaborate with a major technology company. Ultimately, the Teaching Open Source website the students developed was not adopted, but the students' research, design, as well as deliverables influenced its re-development by Red Hat.

## SUGGESTIONS FOR IMPLEMENTATION

In this section the authors offer additional considerations for instructors or partner organizations preparing to undertake a student learning experience that will use open or inner source development practices:

- **Organization buy-in:** Success at short-term projects of a semester length requires that companies actively embrace the opportunity for open exploration of a business challenge or problem with a university and its students. Instructors who are uneasy about the potential for a single client failure might instead consider working with an established open source project team at the Free Software Foundation (fsf.org/campaigns/priority-projects), Apache Foundation (www.apache.org), or GitHub (github.com/explore);
- **Organization active contact:** Instructors should look for the client who has interest in the university and program. An experienced industry professional who sits on a college or program advisory board would be a great resource to help in the project or to refer a colleague who would actively engage the students as their trusted mentor;
- **Organization time allocation:** One of the most precious assets a mentor can offer students is his or her time. To do so, a mentor may need to relinquish or delegate other work duties. Of course, an ideal collaboration would let the professional volunteer a substantial amount of his or her work week, but this is not usually possible (as it was in our case). At a minimum, instructors should advocate the project as a short-term commitment allowing the company to build relationships and gain access to undergraduates' talent and skills to increase company participation in projects;
- **Organization project need:** Which OSS projects should the IT industry consider appropriate for collaboration with university partners? The optimal scenario would be a relatively low-risk idea or unfinished project that has not yet warranted much corporate investment or time. If the students fail to deliver a result meeting all client requirements, the organization can continue with the next semester's class, recruit interns from the students, or use internal resources to finish it;
- **Course time and scope fit:** Project success depends on the student team being given the time and resources to succeed. The aforementioned project, for example, dominated a fifteen-week semester and demanded half of the course evaluation. If the university calendar is short, or the coursework cannot accommodate a large project, the authors suggest reducing the project scope. Consider improving an aspect of an existing open source project. Many projects need help with documentation, particular feature sets, etc. SourceForge (sourceforge.net/p/forge/helpwanted/) provides a listing to start the process.

## CONCLUSION AND FUTURE DIRECTIONS

This paper describes how student-led collaboration with a large technology provider enhanced the learning experience for an undergraduate web design and development class. Instead of small teams of students using limited deliverables to design websites for local businesses or nonprofits (as in previous semesters), a decision to "open source the pedagogy" challenged the class to collaborate on a more substantial project, building a website for an S&P 500 company.

Success, however, required major classroom changes to accommodate the open source principles of cooperation, collaboration, and meritocracy, as well as Agile project management. To replicate this active learning opportunity, instructors must be willing to shift their primary responsibility from presenting course content to becoming the facilitator of a learning process that may exceed their comfort zone. The client must be able to provide a mentor to work with students for the duration.

Similarly, students must forgo the familiarity of exams and predetermined deliverables, as well as their direct correlation to a final grade, while gaining considerable control of their deliverables and the final project outcome. Accordingly, they participate in intense teamwork, new technologies,

and daily checkpoints and communication. Students may not fully perceive the real-world benefits of such an undertaking until they seek internships and jobs. However, learning projects based on open or inner source principles do prepare students for today's workplace.

Future collaboration between open source and academic realms is promising, and the authors want to see such initiatives become fundamental to undergraduate coursework. They continue to develop the best practices applicable to replicating the experience described herein.

## REFERENCES

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Thomas, D. (2001). *Manifesto for Agile Software Development*. Retrieved June 21, 2016, from http://agilemanifesto.org/

Capraro, M., & Riehle, D. (2016). Inner Source Definition, Benefits, and Challenges. *ACM Computing Surveys*, *49*(4), 36. doi:10.1145/2856821

CATME. (2019). Retrieved on February 13, 2019 from http://info.catme.org/

Coldwell, J., Craig, A., & Goold, A. (2011). Using eTechnologies for Active Learning. *Interdisciplinary Journal of Information, Knowledge, and Management*, *6*, 95–106. doi:10.28945/1367

Drake, J. (2012). A Critical Analysis of Active Learning and an Alternative Pedagogical Framework for Introductory Information Systems Courses. *Journal of Information Technology Education: Innovations in Practice*, 39–52.

Edwards, D. (2012, September 21). *The History of DevOps*. Retrieved from http://itrevolution.com/the-history-of-devops/

Hat, R. (2009). *The Open Source Way*. Retrieved from http://www.theopensourceway.org/book/index.html

InnerSource Commons. (n.d.). Retrieved February 13, 2019, from https://paypal.github.io/InnerSourceCommons/

Innersourcing. (n.d.). Retrieved February 13, 2019 from http://www.inner-sourcing.com/

Izquierdo-Cortazar, D., González-Barahona, J., Robles, G., Deprez, J., & Auvray, V. (2012). *FLOSS Communities: Analyzing Evolvability and Robustness from an Industrial Perspective*. Retrieved from https://hal.inria.fr/file/index/docid/1058788/filename/izquierdo-cortazar-etal.pdf

Kamthan, P. (2007). On the Prospects and Concerns of Integrating Open Source Software Environment in Software Engineering Education. *Journal of Information Technology Education*, *6*, 45–64. doi:10.28945/201

Koohang, A., & Harman, K. (2005). Open Source: A Metaphor for E-Learning. *Informing Science Journal*, *8*, 75–86. doi:10.28945/488

Kruck, S. E., & Teer, F. P. (2009). Interdisciplinary Student Teams Projects: A Case Study. *Journal of Information Systems Education*, *20*(3), 325–330.

Lakhani, K. R., & von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy*, *32*(6), 923–943. doi:10.1016/S0048-7333(02)00095-1

Layton, R. A., Loughry, M. L., Ohland, M. W., & Ricco, G. D. (2010). Design and validation of a web-based system for assigning members to teams using instructor-specified criteria. *Advances in Engineering Education*, *2*(1), 1–28.

Long, J. (2009). Open Source Software Development Experiences on the Students' Resumes: Do They Count? - Insights from the Employers' Perspectives. *Journal of Information Technology Education*, *8*, 229–242. doi:10.28945/618

McComb, S., Green, S., & Compton, W. (1999). Project goals, team performance, and shared understanding. *Engineering Management Journal*, *11*(3), 7–12. doi:10.1080/10429247.1999.11415033

O'Mahony, S., & Ferraro, F. (2007). The Emergence of Governance in an Open Source Community. *Academy of Management Journal*, *50*(5), 1079–1106. doi:10.5465/amj.2007.27169153

O'Reilly, T. (2000). *Response to Matt Feinstein on Open Source and OpenGL*. Retrieved June 25, 2016, from http://archive.oreilly.com/pub/a/oreilly/ask_tim/2000/opengl_1200.html

Open Source Initiative (OSI). (2019). *Licenses & Standards*. Retrieved January 31, 2019, from https://opensource.org/licenses

Poindexter, S. (2003). Assessing Active Alternatives for Teaching Programming. *Journal of Information Technology Education*, *2*, 257–265. doi:10.28945/326

Pundak, D., Herscovitz, O., Shacham, M., & Wiser-Biton, R. (2009). Instructors' Attitudes toward Active Learning. *Interdisciplinary Journal of E-Learning and Learning Objects*, *5*, 215–232. doi:10.28945/74

Raymond, E. S. (2001). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media, Inc.

Sack, W., Détienne, F., Ducheneaut, N., Burkhardt, J.-M., Mahendran, D., & Barcellini, F. (2006). A Methodological Framework for Socio-Cognitive Analyses of Collaborative Design of Open Source Software. *Computer Supported Cooperative Work*, *15*(2-3), 229–250. doi:10.1007/s10606-006-9020-5

Sahin, Y. (2011). A team building model for software engineering courses term projects. *Computers & Education*, *56*(3), 916–922. doi:10.1016/j.compedu.2010.11.006

Schiller, S. Z. (2009). Practicing Learner-Centered Teaching: Pedagogical Design and Assessment of a Second Life Project. *Journal of Information Systems Education*, *20*(3), 369–381.

Stallman, R. (1992). *Why Software Should Be Free - GNU Project - Free Software Foundation*. Retrieved June 11, 2016, from https://www.gnu.org/philosophy/shouldbefree.html

Stewart, K. J., & Gosain, S. (2006). The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *Management Information Systems Quarterly*, *30*(2), 291–314. doi:10.2307/25148732

Stol, K. J., Avgeriou, P., Babar, M. A., Lucas, Y., & Fitzgerald, B. (2014). Key factors for adopting inner source. *ACM Transactions on Software Engineering and Methodology*, *23*(2), 1–35. doi:10.1145/2533685

Stol, K. J., Babar, M. A., Avgeriou, P., & Fitzgerald, B. (2011). A comparative study of challenges in integrating Open Source Software and Inner Source Software. *Information and Software Technology*, *53*(12), 1319–1336. doi:10.1016/j.infsof.2011.06.007

Stol, K. J., & Fitzgerald, B. (2015). Inner Source–Adopting Open Source Development Practices in Organizations: A Tutorial. *IEEE Software*, *32*(4), 60–67. doi:10.1109/MS.2014.77

Teel, S., Schweitzer, D., & Fulton, S. (2012). Teaching Undergraduate Software Engineering Using Open Source Development Tools. *Issues in Informing Science and Information Technology*, *9*, 63–73. doi:10.28945/1604

Turnu, I., Melis, M., Cau, A., Setzu, A., Concas, G., & Mannaro, K. (2006). Modeling and simulation of open source development using an agile practice. *Journal of Systems Architecture*, *52*(11), 610–618. doi:10.1016/j.sysarc.2006.06.005

Uzunboylu, H., Bicen, H., & Cavus, N. (2011). The efficient virtual learning environment: A case study of web 2.0 tools and Windows live spaces. *Computers & Education*, *56*(3), 720–726. doi:10.1016/j.compedu.2010.10.014

von Krogh, G., Haefliger, S., Spaeth, S., & Wallin, M. W. (2012). Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *Management Information Systems Quarterly*, *36*(2), 649–676. doi:10.2307/41703471

*Alan Rea is a Professor of Computer Information Systems in the Department of Business Information Systems at the Haworth College of Business, Western Michigan University. Teaching courses in Information Security and Object-Oriented Programming, Dr. Rea integrates free and open source software and whenever possible, agile approaches, to accommodate the dynamic environment within information systems. His research concentrates on secure application and system development as well as organizational information assurance and risk management approaches. In particular, he has examined security and privacy implications associated with developing, deploying, and managing web and mobile applications as well as Internet of Things devices. His research has been published in the Journal of Information Systems Education, International Journal of Electronic Healthcare, Journal of Information Systems Security, Journal of Information Privacy and Security, Journal of Computer Information Systems, Communications of the ACM, and Journal of Digital Forensics, Security and Law. Currently he co-directs a cross-disciplinary M.S. and Graduate Certificate in Information Security.*

*Nick Yeates is an Open Source Strategy Consultant, having recently worked at Red Hat Inc to lead their customers through the open source way. He is a member of the InnerSource Commons community, working to formalize the art, science, and psychology needed to bring open source culture into organizations. Mr. Yeates has a community management background, as well as technical experience from mature startups, higher education, and military defense agencies. At Zenoss Inc, he led their online open source community toward a distributed software development paradigm. At the U.S. DoD (Department of Defense), he led a program-wide InnerSource implementation His interests lay at the intersection of open technology, people, and business. Mr. Yeates enjoys bringing transparent, communal, and iterative cultures into any organization.*