**Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS)[[1]](#footnote-1)**

xx

Heidi J. C. Ellis, Western New England University

Gregory W. Hislop, Drexel University

STONEY Jackson, Western New England University

Lori Postner, Nassau Community College

Providing students with the professional, communication and technical skills necessary to contribute to an ongoing software project is critical, yet often difficult in higher education. Involving student teams in real-world projects developed by professional software engineers for actual users is invaluable. Free and Open Source Software (FOSS) has emerged as an important approach to creating, managing, and distributing software products. Involvement in a free and open source software project provides students with experience developing within a professional environment, with a professional community, and has the additional benefit that all communication and artifacts are publicly accessible. Humanitarian Free and Open Source Software (HFOSS) projects benefit the human condition in some manner. They can range from disaster management to microfinance to election monitoring applications. This article discusses the benefits and challenges of students participating in HFOSS projects within the context of undergraduate computing degree programs. This article reports on a multi-year study of students' self-reported attitudes and learning from participation in an HFOSS project. Results indicate that working on an HFOSS project increased interest in computing. In addition, students perceive that they are gaining experience in developing software in a distributed environment with the attendant skills of communication, distributed teamwork and more.

**INTRODUCTION**

Free and Open Source Software (FOSS) is developed under licenses that provide users with the right to run the software for any purpose, modify, and distribute the software for free. FOSS is usually created and maintained by teams of designers, developers, and people with other IT skills who contribute to the project. FOSS contributors typically include a mix of volunteers and people paid to work on the project by a company or foundation. FOSS has become main-stream with applications such as Firefox, Apache, MySQL, Eclipse and Android to name a few. Older FOSS products such as GNU/Linux have been in widespread use for many years, with use expanding rapidly in business in recent years [Noyes 2011].

FOSS culture has several characteristics that lend itself to student participation:

* **Communal Development:** Many FOSS projects are community-based meaning that the community decides the direction of the project. This community provides students with a group of professionals to interact with and from which to learn. Since communication is crucial in maintaining a healthy community, FOSS projects provide numerous opportunities for students to learn about professional communication.
* **Openness**: Most open source projects welcome new contributors including volunteers. While there are some FOSS projects that do not seek new contributors, many FOSS projects are eager to embrace students as they see students as possible long-term contributors who will help sustain the project in the long run.
* **Transparency:** All of the source code and other artifacts are available. The code and artifacts are also modifiable and long-lived, allowing the history of the project to be examined. This transparency allows students to access artifacts throughout the project history as well as understand who contributed to various portions of the project. In addition, this transparency provides students with the opportunity for their contributions to be visible.
* **Open Licensing:** Open source software is distributed under a license that indicates that the source code must be available for modification. This licensing avoids the issues of intellectual property and proprietary development that can occur with student involvement in industry projects.
* **Distributed, Global Environment:** Many FOSS projects are both used and developed around the world, providing students a chance to observe and participate in a globally distributed software team.
* **Meritocratic Process:** FOSS operates on a form of meritocracy where responsibilities increase as a contributor demonstrates his or her capabilities. This environment allows students to gain confidence in their abilities as well as encouraging them to grow professionally.

These central FOSS principles provide unique opportunities for students to observe and participate in many aspects of software work and thus provide excellent educational opportunities. As a result, educators have increasingly adopted student involvement in FOSS projects to provide students with professional experience with a real-world project [Marmostein 2011, Ellis, Hislop, Chua, and Dziallas 2011, Ludi 2011].

Humanitarian FOSS is FOSS that somehow benefits the human condition. HFOSS applications range from disaster management to medical records to education to microfinance and more. HFOSS has all of the characteristics of FOSS and the added benefit that they are altruistic in nature. Some studies have shown that the aspect of “doing good” is attractive to students [Homkes 2008, Hislop, Ellis, and Morelli 2009, Ellis, Hislop, Rodriguez, and Morelli 2012]. The altruistic nature of the project seems to make the communities more welcoming to new contributors and more helpful to student participants. HFOSS was used in this study as the real-world and humanitarian purpose of HFOSS is well aligned with characteristics that research has shown attracts students, and particularly women, to computing majors [Beyer, Rynes, and Haller 2004, Carter 2006, Cohoon 2002, Tillberg, and Cohoon 2005].

**HFOSS in Education**

This section provides a broad description of the use of HFOSS in education. It includes the ways that HFOSS may be used to support student learning, the benefits to student learning, and the roadblocks to student participation. The authors believe that this discussion is generalizable to most FOSS projects.

**Uses of HFOSS in Education**

HFOSS can be used to support student learning on a variety of topics, with a variety of learning experiences, and in a variety of venues. In some cases, the HFOSS project may provide an artifact of study with minimal or no interaction with the HFOSS community while in other cases students may become members of the HFOSS community and contribute to the community as part of a class or other academic activity.

The variety of learning topics that HFOSS can support is large [Morelli and deLanerolle 2009, Kussmaul, Ellis, and Hislop 2012]. When people first consider student contribution to HFOSS, they tend to think of code contributions. It is essential to remember that open source projects perform the full range of software engineering activities, and do so transparently. This provides students the opportunity to interact with development professionals and to work on non-code aspects of projects such as specification, planning, documentation, support, testing, etc. where non-coding skills can be observed and practiced. Students may also learn about non-technical aspects of HFOSS by performing product reviews, investigating the use of a tool within an HFOSS project, exploring the social norms of communication within an HFOSS community and more. These activities may be individual efforts or carried out by a team.

In addition to providing a range of learning topics, the range of learning activities is also wide. HFOSS can be incorporated into courses at a variety of levels including:

* A single assignment – e.g., verify a bug
* A sequence of assignments – e.g., verify a bug, fix the bug, write test cases for the fix
* A project deliverable – e.g., create or update a software requirements specification
* A capstone project – e.g., add a feature to an application, test, and update all relevant documentation
* As the subject of a course - e.g., a course on the culture and tools of HFOSS

The venues that support student learning via participation in an HFOSS project also vary. The obvious place for student learning is the classroom, but HFOSS can also be used for independent studies, internships, hackfests, clubs, and other non-class environments. Some institutions have a “makerspace” where students can congregate and work on projects of their choosing.

Lave and Wenger [Lave and Wenger 1991, Wenger 1999] use the term *community of practice* to describe a group of people engaged in a common task or who have a common goal. Many FOSS practitioners recognize communities of practice as a model that rings true for their work [Meiszner, Glott, and Sowe 2008]. Student learning in an HFOSS community is a form of legitimate peripheral participation [Lave and Wenger 1991] where novices are introduced to the community of practice in steps. Initially, a new member participates peripherally through watching, listening and observing. The novice progresses to legitimate peripheral participation in the community through a small but meaningful interaction and eventually becomes a member. This approach is already recognized as a valid learning approach in some FOSS projects [Ye and Kishida 2003]. Student engagement in an HFOSS community of practice is a form of cognitive apprenticeship [Collins 2006] where students have the opportunity to learn from “masters of the craft”. However, unlike many cognitive apprenticeship models [Brown, Collins, and Duguid 1989] where instructors scaffold the students' activities, student learning in HFOSS will be based on the expertise of the open source community of practice. Learning within the community provides a number of advantages including providing students with a wider range of learning resources than a classroom environment, a built-in support system, motivation, and more [Meiszner, Glott, and Sowe 2008].

**Benefits to Student Participation in HFOSS**

Student learning via participation in HFOSS holds benefits for students, instructors, and academic institutions, in addition to directly benefiting the HFOSS project. These benefits include:

**Technical Skills** – Gaining a range of technical skills is one obvious benefit of student involvement in HFOSS. Such skills include programming, testing, debugging and more. However, many skills that students gain by working on an HFOSS project are often not taught in the classroom including code maintenance, version control, and development tools [Exter 2014].

**Professional Skills** – In addition to technical skills, students gain skills necessary to operate within a professional environment. Such skills include communication, critical thinking, problem solving, team building, and continued learning. Glott et. al. identified a wide range of learning via student participation in FOSS communities including professional skills, patents and licensing, management skills, coding skills and more [Glott, Andreas, Sulayman et al 2007]. Students also often have opportunities to learn communication norms and to observe good and bad examples of professional communication.

**Learning within a Professional Community** – When students participate in an HFOSS project, they are practicing a form of cognitive apprenticeship, learning from the experts in the HFOSS community [Collins 2006]. Learning within the community provides a wider range of learning resources than may be found in a traditional classroom as well as a support system comprised of experts with a range of backgrounds [Meiszner, Glott, and Sowe 2008]. In addition, FOSS projects can support a variety of learning techniques including active learning, problem-based learning, and collaborative learning [Weller, Meizsner, Sowe, and Karoulis 2008]. HFOSS holds the potential for broad professional learning.

**Distributed Development** – HFOSS projects are usually distributed development environments with a global community. Such development provides a unique opportunity for students to interact with professionals in a variety of locations from a variety of cultures to develop software, preparing students to interact in distributed professional environments. This global development is true for open source in general, but perhaps more so for humanitarian projects where the client base is often in the developing world while technical contributions are spread across developing and developed countries.

**Project Complexity –** Many HFOSS projects are of a significant size and complexity which allows students to gain an understanding of how size and complexity impact development. Students gain professional experience in large, complex, real-world environments that cannot be easily reproduced in traditional classrooms. In addition, HFOSS projects are ongoing with a multi-year lifespan. This allows students to better understand maintenance and the impact of decisions made throughout the product’s lifespan.

**Software Process -** HFOSS projects typically use a very agile development process which students must learn in order to participate in the project. The transparent and distributed team approach of HFOSS projects requires flexible, tool-based project environments that are centered on the Web. This tends to make FOSS projects early adopters of many development approaches (revision control, continuous integration, bug and feature trackers, etc.). In fact, many of the well-known products in these areas are open source projects themselves (Git, Hudson, Bugzilla, etc.).

**Social Awareness -** By working on software that benefits the human condition, students gain an understanding of social responsibility. In addition to conveying knowledge, educational institutions desire to prepare students to be contributing members of the community as well as good citizens. Participation in HFOSS allows students to directly see the benefits of using their computing skills to help others.

**Motivation** – A series of studies indicates that students are motivated by making contributions to HFOSS projects that have a real client base. The humanitarian purpose of HFOSS projects seems particularly effective at providing motivation [Hislop, Ellis, and Morelli 2009, Ellis, Hislop, Rodriguez, and Morelli 2012]. Other studies indicate that seeing the positive impact of IT is particularly useful in attracting women and other under-represented groups to computing majors.

**Resume building** - Since HFOSS projects are transparent, student contributions are documented and can be easily referenced. Students can provide potential employers with pointers to their work and demonstrate their ability to work within a real-world project.

In addition to the benefits that students gain, instructors also benefit by having their students become involved in HFOSS projects. As an instructor, it is often difficult to keep up with the latest trends and innovations. The HFOSS community supplies a group of practicing professionals with a wide range of knowledge. This group can help answer student questions and support instructor learning which allows students to learn the most recent tools and techniques without putting an enormous stress on the instructor. By collaborating with an HFOSS project, instructors can stay current with the approaches and tools used in the professional environment. A less tangible benefit to instructors is the feeling of accomplishment in producing students that are prepared to enter the professional environment as well as the knowledge that the efforts of students have “done good”.

Lastly, there are multiple benefits to academic institutions and the community as a whole to student participation in HFOSS. First, increased social awareness in students has the potential to create a more collaborative environment on campus. In addition, many institutions include some aspect of contributing to the community as part of their mission. Student participation in HFOSS is one unique way to fulfill this mission. Lastly, HFOSS collaboration may involve multiple academic institutions, allowing significant resources to be applied to social needs. This allows the academy to positively impact society at a global level.

**Challenges to Student Participation in HFOSS**

While there are significant potential benefits from student participation in HFOSS, there are also significant challenges for students, instructors, and academic institutions that need to be addressed to achieve these benefits.

One major challenge to students is adjusting to a less-structured learning environment. Students who enter a class with the expectation that learning will be structured and predictable as it has been in many of their prior classes may have difficulties adjusting to the open learning that occurs within an HFOSS project. This is not to say that learning within an HFOSS environment is random, but that the student needs to learn from the variety of resources provided by the HFOSS community including documentation, listserv logs, IRC meetings, as well as by interacting with the professional community. Students' expectations about the role of the instructor must be altered; the teacher is no longer the authority for the material, the community is.

Students face a number of learning curves upon entering an HFOSS project. Students may not be familiar with the tools, approaches or language used in the project and students may need to learn all of the above simultaneously. For instance, students may need to learn both the principles and reasons for version control while using git and coding in a new language. The complexity of the project itself presents another learning curve as students must learn how to approach and understand a project that may have tens of developers and a half a million lines of code. Students face yet another learning curve to understand FOSS culture in order to participate in an HFOSS project. Issues such as forms and modes of communication must be learned in addition to software process. Lastly, students may need to learn significant domain knowledge to work on a project, yet another learning curve.

Faculty members are also faced with challenges when involving students in HFOSS projects within the classroom. They may face similar learning curves for tools, approaches, and FOSS culture, as well as for domain knowledge. In addition, instructors must also foster student participation. They must learn how to create assignments and course deliverables as well as create grading rubrics for these. They must negotiate communication and support with the HFOSS community. For instance, identifying what exactly the community will provide. This could be a mentor for the instructor, someone to answer student questions, wiki space or other support. The instructor needs to make clear what the students will provide, which could include documentation, code fixes, code enhancements and more. Instructors also need to be mindful of potential FERPA and intellectual property issues.

Project selection is critical when incorporating HFOSS into a course, and may be a significant challenge. It is important that the project fit the learning goals of the course, be open to contributions and welcoming to students, and have good communication mechanisms. Fortunately, the initial effort can benefit learning activities across a series of courses, not just a single course offering. Several efforts have reported on approaches to FOSS project selection for use with students [Gokhale, Smith, and McCartney 2012, 2014, Smith, McCartney, Gokhale, and Kaczmarczyk 2014, Ellis, Purcell, and Hislop 2012].

The issues presented above mostly apply to instructors supporting student participation in HFOSS for the first (or second) time. However, there are a number of faculty challenges that are inherent in the environment and occur in both initial and subsequent terms. Synchronizing schedules between the academic calendar and the HFOSS release cycle may present problems if students are contributing code. The unpredictability also presents a challenge where the project may make a major turn mid-term. For instance, a project may decide that it no longer needs the contributions planned for students, leaving the instructor to figure out how to manage contributions that may not get accepted into the community. Instructors also need to plan how to gracefully end a term so that the HFOSS community understands what is done and what is undone when students are done with a term.

The challenges faced by institutions are somewhat less significant. Institutions need to be able to support the HFOSS effort by providing access to GNU/Linux (typically) and open source applications in student labs. However, the benefits to the institution greatly outweigh the infrastructure they need to supply. The potential positive impact of humanitarian projects is an easy fit to the mission of most institutions of higher education.

**Related Work**

Open source software has been used classrooms in at least three ways. One use is to take FOSS artifacts as objects of study in the classroom. Examples include studying code to discover design patterns and making local enhancements to code that are not contributed back to the FOSS project. A second use is for institutions to develop their own FOSS project. For example, students may work on a course registration system for their institution that will be distributed as FOSS. A third is direct student participation in an existing FOSS community. In this case, students contribute directly to the community. All three approaches are described in this section. In addition, a discussion of student participation in HFOSS is also included.

**FOSS as an Artifact**

Open source software has been used as an object of study in computing classes since at least the early 2000’s [Carrington and Kim 2003]. Nascimento et. al. explored how FOSS projects have been used in software engineering education [Nascimento, Cox, Almeida, Sampaio, Almeida Bittencourt, Souza, and Chavez 2013]. They discovered that a number of software engineering courses used open source software either as a source of study or to support student involvement. Some courses focused on specific areas of software engineering such as Design or Maintenance.

Chavez et. al. describe a myriad of ways that FOSS can be utilized within software engineering courses and report on how FOSS is used within courses as an object of study as well as supporting Brazilian student participation in FOSS communities [Chavez, Terceiro, Meirelles, Santos, and Kon 2011]. The authors identify a variety of possible ways that FOSS may be used in the classroom ranging from a study of FOSS culture to contributing code within a software engineering course. The authors also highlight the challenges to involving students in FOSS projects for instructors. These challenges include helping students with FOSS tools that may be unfamiliar, mentoring students on FOSS culture and matching required course content with project content.

FOSS provides many opportunities for students to learn about computing by studying FOSS artifacts. Nandigam, Gudivada, and Hamou-Lhadj [2008] used examination of source code from various popular projects such as Apache, Hibernate and JUnit to convey software engineering topics including coding conventions, coding style, design quality, reverse engineering, refactoring, assessing the impact of changes. The authors reported some roadblocks in getting the tools working, but that students gained a better understanding of software engineering concepts through this process.

McCartney, Gokhale, and Smith [2012] describe an undergraduate software engineering course that reverse engineers and modifies FOSS projects. Results of a paired pre- post-course survey indicate that students gained an understanding of maintenance and the need for documentation. Results also suggest that students may have gained an understanding of the value of reverse engineering, as well as tool use. Costa-Soria and P´erez [2009] describe a similar course that reverse engineers FOSS projects to determine their software architectures and to explore aspect-oriented software development. The goal is to help students develop skills in abstraction and design. The approach includes analysis, design and maintenance phases.

In addition to observing FOSS artifacts, there are several courses where students make a contribution or enhancement to a project. In the cases described here, these contributions are used for learning purposes only and are not contributed back to the community. Toth [2006] describes an approach to teaching a software engineering practicum course at Portland State University that revolves around extending existing FOSS tools such as Eclipse. Toth’s approach describes a two-term practicum where the tools and artifacts developed are fed back into the main software engineering courses. Lessons learned from this effort include that open source tools such as Eclipse may have a steeper learning curve than equivalent proprietary tools and that there is also a significant learning curve when extending open source tools.

Bucheta et. al. describe an advanced software engineering course that taught software evolution while providing a professional experience by having students fulfill change requests for common FOSS projects such as Notepad++ [Buchta, Petrenko, Poshyvanyk, and Rajlich 2006, Petrenko, Poshyvanyk, Rajlich, and Buchta 2007]. During the course, students implement three change requests. Student survey results indicate that students rated the course more highly and indicated that they learned more than when the course was taught without the open source approach. Xing [2010] reports on a graduate course where students developed enhancements for a FOSS project. Survey results indicated that students enjoyed working on a FOSS project and that working on a FOSS project is engaging. Students also reported that participation in a FOSS project is an effective way to learn.

There are multiple benefits to be reaped from the study of FOSS artifacts. The transparency of FOSS projects provides students with many artifacts to study as well as providing an existing code base to extend. Instructors have great freedom in selecting both the artifact and the assignment and the detachment from the FOSS community provides flexibility in scheduling and deliverables. However, there are some drawbacks to this approach. Students do not get the opportunity to learn from the community. They do not observe the professional norms and expectations. In addition, they do not get the motivation from seeing their contributions being accepted into a real-world project and they cannot use their work to build their resume.

**Locally Developed FOSS Projects**

In addition to using an existing project, many instructors create an in-house FOSS project for students to work on. This approach has the advantage that the instructor has more control over development including deliverables, schedules, and more. However, this approach lacks the community support and involvement in a live professional community seen when involving students in ongoing FOSS projects.

Santore et. al. [2010] describe a software engineering class that developed a GUI front-end to Subversion. The students in the course had not yet taken upper-level courses. The authors report that students were motivated by working on a real-world project. Auer, Juntunen, and Ojala [2011] describe the OpixProject, a FOSS project for project management developed at Oulu University of Applied Sciences (OUAS). The project was created to provide Master’s students with a real-world experience, including gaining professional and technical skills within a more controlled environment than found in most real-world projects.

There are many examples of service-learning experiences where students develop software for non-profit organizations [Bloomfield, Sherriff, and Williams 2014, Goldweber et. al. 2011, Venkatagiri 2006, Carter 2011, Stone and Madigan 2011, Cicirello 2013, Sabin 2011]. These organizations are typically local so that students can interact with the client, although some efforts, Goldweber et. al. for example, create global projects. Students learn about real-world issues when working on these service learning projects and also gain technical knowledge. However, there is typically no supportive professional community with which the students can interact and the project is built from scratch rather than building on an existing system. Students do not gain experience in reading and understanding code or with interacting with practicing professionals.

**Student Involvement in the FOSS Community**

As discussed in section 2 above, direct student participation in FOSS has the benefits of providing a real-world project and professional environment in which to learn. There are many efforts that describe student participation in FOSS projects, however little study has been done on the actual learning that occurs within the environment. In this subsection we highlight some of the efforts to involve students in FOSS and identify those with any sort of research results.

Some of the early efforts in fostering student participation in FOSS projects occurred at the Master’s level. It makes sense that the initial experimental efforts in student involvement in FOSS would occur with more mature students who could be expected to have stronger technical skills. Jaccheri and Osterlie [2007] discuss experiences in involving Master’s students at Norwegian University of Science and Technology in FOSS projects. Jaccheri and Osterlie emphasize that FOSS should be considered as a social discipline that involves collaboration and that it is important to be connected to the community. They also noted that the complexity of FOSS projects can pose difficulties for students and that allowing students to choose their own project results in increased motivation. In a similar effort, Lundell, Persson, and Lings [2007] report that Master’s students in a Open Source and Distributed Development Models course at the University of Skövde, Sweden were required to contribute to a FOSS project. In this case, student contributions did not have to be accepted by the community, the student only had to make an effort to contribute and to interact with the community. The authors report that the course was one of the most popular in the program. Martinez [2009] reports on an online Master's program at the Universitat Oberta de Catalunya where students must either found a new FOSS project or join an existing project. The article describes Martınez’s successful personal experience with contributing to GNOME.

The involvement of students in FOSS is starting to permeate undergraduate education as well. Marmostein [2011] reports on an effort to involve students in a FOSS community and have them make a small contribution to the project. Results of student opinion surveys indicate that students felt that they learned software tools and communication skills. Interestingly, students did not report learning team skills, perhaps due to the fact that the FOSS community in which they participated had unresponsive members. Gehringer [2011] describes options of FOSS project managers on best practices for faculty members desiring to involve students in FOSS projects. Gehringer also describes several projects to which students have made contributions.

Kussmaul [2009] presents a USABL model for gradually getting students involved in FOSS projects. The model includes steps to **U**se the FOSS product to accomplish a task, **S**tudy the software as an example, **A**dd a minor enhancement, **B**uild a larger component and **L**everage the FOSS project to solve other problems. In this manner, students are gradually introduced to participating in FOSS projects starting with simple tasks and building to the more complex. Anecdotal data from the experience indicates that students learned about real projects for real customers.

Morgan and Jensen [2014] describe two courses that involve students in FOSS projects. The article compares a course that uses the communities of practice model with a course taught using a more traditional classroom approach. While both approaches were successful, Morgan and Jensen indicate that the communities of practice approach appeared to create a sense of community within the classroom that helped students when learning within the unstructured environment of FOSS. Morgan and Jensen mention challenges that include project selection, creating assignments, grading, and more.

There are also two notable efforts that use multi-institutional teams. The Undergraduate Capstone Open Source Project (UCOSP) is a widely known course that brings together undergraduates from across Canada and sometimes the U.S. to work on a FOSS project with oversight by a mentor who is either a faculty member or a member of the FOSS project [Stroulia, Bauer, Craig, Reid, and Wilson 2011]. Teams of five to eight undergraduates from different institutions work on a real-world project within the FOSS community. Benefits to students from involvement with UCOSP include learning new programming languages and tools, gaining professional experiences, and professional networking. In a similar effort, in November 2013, Jay Borenstein teamed up with Facebook to create the Open Academy Program which is intended to provide students with real-world experience via participating in a FOSS project [Chacon and Borenstein 2014, https://www.facebook.com/OpenAcademyProgram]. Students work as part of a multi-institutional team with mentorship by industry professionals to contribute to an ongoing FOSS project. The program includes Stanford, MIT, Cornell and others.

**Student Involvement in HFOSS**

The number of efforts related to involving students in a Humanitarian FOSS project (HFOSS) are somewhat fewer than for the larger collection of FOSS projects. This makes sense given that HFOSS represents only a subset of the number of FOSS projects under development.

Liu [2005] presents an early effort to involve students in HFOSS projects via service learning. He suggests an open source software development process to support student involvement in HFOSS projects. The process includes milestones, evaluation criteria for projects, and document and coding style guidelines. Ding [2007] discusses a virtual service learning model that involves students in FOSS projects to aid in learning of professional communication and documentation. While the FOSS projects were not just humanitarian, the effort was formulated as a virtual service learning experience for students. Interaction with the professional community provides the base for such learning.

Another interesting effort is related to involving students in developing games for the One Laptop Per Child project [Jacobs 2010]. The course described is so motivating that students who have graduated mentor current students on their own time. This effort also leads to sponsored research opportunities. MacKellar, Sabin, and Tucker [2013] present an approach to developing real-world HFOSS projects for real clients within an academic environment. The authors report that the approach can be scaled to different institutions, however the problems of students' and instructors' learning curve as well as time constraints still occur.

Morelli, deLanerolle, and Tucker [2012] describe the Humanitarian FOSS effort which was started in 2006. This chapter describes the beginnings of the Humanitarian FOSS project where students were involved in the Sahana disaster management project. The Humanitarian FOSS effort has had students contribute to 16 different HFOSS projects. The work described in this article grew out of the HFOSS project and one of the authors was a founding member of the Humanitarian FOSS project.

**The Study**

This article reports on the results of a six-year study of the impact of student participation in HFOSS projects on motivation and learning. The study is driven by a desire to understand whether student involvement in HFOSS projects is motivating to students as well as to explore student learning within the HFOSS environment.

The study looks at three aspects related to students' participation in HFOSS:

1. **The impact of student participation in HFOSS on attitude towards computing**. In particular, are students motivated by participating in HFOSS and do female students find working with an HFOSS project an attractive aspect of Computer Science?
2. **Perceived learning related to software engineering**. Do students learn software engineering skills by participating in HFOSS? Both technical and professional skills such as teamwork and communication are investigated.
3. **The impact of student participation in HFOSS on major selection and career plans.** Does participating in an HFOSS project cause students to be attracted to computing as a career? Are existing computing students more persistent in their career choice because of their HFOSS participation?

There are three sets of research hypotheses investigated in this study that correspond to the three aspects of interest in the study. Each set has a null and a directional hypothesis:

**H1o**: Student participation in an HOFSS project has no impact on student motivation to study computing.

**H1a**: Student participation in an HFOSS project has a positive impact on student motivation to study computing.

**H2o**: Student participation in an HFOSS project has no impact on perceived learning related to software engineering.

**H2a**: Student participation in an HFOSS project has positive impact on perceived learning related to software engineering.

**H3o**: Student participation in an HFOSS project has no impact on major and career plans.

**H3a**: Student participation in an HFOSS project has positive impact on major and career plans.

The study described in this article involved ten undergraduate courses and internships offered at four different academic institutions in the U.S. between summer 2008 and fall 2013. The institutions involved were typically small, liberal arts institutions with between 1500 – 3000 undergraduates. The courses in the study ranged from introductory courses for non-majors to senior-level software engineering courses, but most were upper-level project-oriented courses. Class sizes were small with 20 or fewer students per class. The study also included three 10-week summer internships. The HFOSS projects included Sahana (disaster management), OpenMRS (medical records), GNOME Accessibility tools, and others. Student involvement ranged from documentation to writing tutorials for tools used by the project to code contributions.

The main study instrument was an anonymous survey that included some background information as well as a series of response statements that used a Likert scale. The background information included name, gender, major and self-reported programming ability. The Likert scale used five points from “strongly disagree” to “strongly agree” with “neutral” as the mid-point value. Options for “Don’t know” and “Not applicable” were also provided. The Likert portion of the survey was divided into three sections, with the first section addressing student perception of motivation, the second addressing student perception of learning, and the third addressing student perception of major and career plans. Pre-course and post-course surveys were administered with no matching of individual pre/post responses (see the Appendix for the complete post-survey). Table 1 shows an example of each of the three types of questions where H1 is an item about motivation, SE1 is related to software engineering learning and O1 is related to major and career. ( “H” is for humanitarian and “O” is for Other.)

Table I. Sample Likert Survey Items about Motivation

|  |  |
| --- | --- |
| Item ID | Description |
| H1 | Working on an HFOSS project gives me a better appreciation for the usefulness of computing. |
| SE1 | I can list the high-level phases that comprise a software project in a real-world environment. |
| O1 | Participation in an HFOSS project may cause me to consider computing as a major or minor. |

The Likert scale responses were converted to ordinal numbers between one and five where one represented “strongly disagree” and five represented “strongly agree”.

**Results**

This section describes the data analysis and discusses the results. The section includes a discussion the central tendency of the results as well as a discussion of the pre- and post-course results. The section begins with an overview of the data.

Table 2 provides an overview of the participation in the survey. It should be noted that in some cases, only post-course surveys were able to be administered. The majority of the surveys were distributed in hard copy form during a class or internship meeting which may have contributed to the high average response rate of 88.82%. In addition, 73% of respondents were male, 13% female and 10% declined to provide gender.

Table 2. Summary of Survey Participation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Semester Offered | Course Type | Administered | Number of Students | Number of Surveys | Response Rate |
| Summer 2008 | Internship | Post | 13 | 13 | 100.00% |
| Fall 2009 | Jr/Sr Course | Post | 17 | 14 | 82.35% |
| Spring 2009 | Jr/Sr Course | Pre | 20 | 20 | 100.00% |
| Spring 2009 | Jr/Sr Course | Post | 20 | 16 | 80.00% |
| Summer 2009 | Internship | Post | 11 | 6 | 54.55% |
| Fall 2009 | Intro Course | Pre | 8 | 8 | 100.00% |
| Fall 2009 | Jr/Sr Course | Pre | 12 | 11 | 91.67% |
| Fall 2009 | Intro Course | Post | 8 | 7 | 87.50% |
| Fall 2009 | Jr/Sr Course | Post | 12 | 11 | 91.67% |
| Summer 2010 | Internship | Post | 15 | 5 | 33.33% |
| Fall 2010 | Sr Course | Pre | 11 | 11 | 100.00% |
| Fall 2010 | Sr Course | Post | 11 | 11 | 100.00% |
| Fall 2011 | Sr Course | Pre | 10 | 10 | 100.00% |
| Fall 2011 | Sr Course | Post | 9 | 9 | 100.00% |
| Fall 2013 | Sr Course | Pre | 6 | 6 | 100.00% |
| Fall 2013 | Sr Course | Post | 6 | 6 | 100.00% |
|  |  | **Total** | **189** | **164** | **88.82%** |

Table 3 shows the breakdown of majors based on the post-course survey. As the surveys were administered within computing courses and internships, it is natural that the majority of the student majors were computer science. It was however interesting to note that some 15% of respondents were unsure of their major. Majors included in the “Other” category included physics, chemistry, biology, economics, English, political science, neuroscience and more.

Table 3. Majors of Respondents - Post

|  |  |  |
| --- | --- | --- |
| Major | Responses | Percent |
| CS | 65 | 65.66% |
| Undecided | 15 | 15.15% |
| Math | 6 | 6.06% |
| Computer Engineering | 3 | 3.03% |
| Other | 10 | 10.10% |

A summary of student ages is shown in Table 4. The ages were evenly distributed between younger students (18-20 years old) and somewhat older students (21-23). There were only a few (5%) students older than 23.

Table 4. Age Distribution - Post

|  |  |
| --- | --- |
| Age Range | Number of Participants |
| 18 – 20 | 46 |
| 21- 23 | 46 |
| 23 or older | 5 |

Figure 1 shows a summary of the students’ perceived programming ability at the end of the course or internship. Programming ability was self-rated by the students with a score between 1 and 5 where 1 indicated a complete novice and 5 was expert. The median self-assessed programming ability was three, while the mode was four.

Fig. 1. Programming Ability – Post.

**Summary of Post-Course Responses**

In this section, we provide an overview of the post-course responses to provide a retrospective on student opinion of participating in an HFOSS project. A high-level observation of the data indicates that all of the Likert items in the survey show that the average student rating for all of the motivation and software engineering items was above the **neutral** rating of three. The results for the items on major/career were mixed.

When looking at the survey items related to impact on student motivation from participation in an HFOSS project, the median for all eleven items was four. The mode was four for all but one item: H2: “I have a greater awareness of the potential for computing to benefit society due to working on an HFOSS project,” for which the mode was 5. When looking at the items with which the largest number of students either disagreed or strongly disagreed, item H8 “Working on an HFOSS project increases my interest in computing” and item H10 “Participating in an HFOSS project made me more comfortable with computing” had the strongest responses with 13.3% disagreeing for item H8 and 12.2% for item H10. As the majority of students were computer science majors, it could be that many students already had a high interest in computing and did not feel that participation in HFOSS increased it. As for H10, one possible explanation for some students feeling that participation in HFOSS did not increase their comfort with computing may come from the characteristics of the project. For most students, working on an HFOSS project was their first exposure to a large, real-world software project and they may have been somewhat intimidated by the size and complexity of the project and its’ development. Therefore their comfort level actually went down because they had a better understanding of the challenges of large scale projects.

The survey items related to software engineering showed a mean and mode of four for all but one item: SE 10 “I am confident that I can maintain an HFOSS project” which had a median and mode of three. There were only two items with which more than 10% of the students disagreed or strongly disagreed. Item SE 10 (11.24%) and item SE 1 “I can list the high-level phases that comprise a software project in a real-world environment” (14.58%). This disagreement may again reflect students’ first exposure to a large scale project.

A more mixed picture is presented by the responses to survey items related to major and career. Table 5 provides an overview of these results. The relatively high disagreement level with item O1 may come from the fact that the majority of students were already declared as computer science majors and therefore were not considering whether to choose it as a major. The high disagreement and low median and mode values for item O3 are actually a positive as this item is phrased in the negative. It could be read that 75% of students have not reconsidered making computing their major due to participation in an HFOSS project. The high disagreement level with item O6 may be due to the domain of the HFOSS projects within which students were involved. Students may be unfamiliar with humanitarian software and therefore there may have been a steep learning curve for the project domain.

Table 5. Overview of Survey Items Related to Major and Career

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item ID | Description | Media | Mod | % |
| O1 | Participation in an HFOSS project has caused me to consider computing as a major or minor. | 3 | 4 | 28.57% |
| O2 | Participation in an HFOSS project has positively reinforced my decision to make computing my major. | 4 | 4 | 10.53% |
| O3 | Participating in an HFOSS project has caused me to question my decision to make computing my major. | 2 | 1 | 75.00% |
| O4 | Participation in an HFOSS project has caused me to consider taking further computing courses. | 4 | 3 | 16.25% |
| O5 | The subject matter of this HFOSS project is highly relevant to my future career plans. | 3 | 3 | 18.82% |
| O6 | I have a high level of experience in the HFOSS subject matter. | 3 | 2 | 41.57% |
| O7 | Overall, I am very satisfied with my learning in the HFOSS project. | 4 | 4 | 5.62% |

The two positive items shown in Table 5 were O2 and O7. The positive medians and modes for these items appear to indicate that the majority of students find participating in an HFOSS project a positive learning experience.

**Pre/Post Test Comparisons**

One important aspect to consider was the change in student opinion from the beginning of the course to the end. As the specific pre- and post-course surveys could not be matched for individuals, the analysis compared the mean results of the pre- and post-course surveys. Table 6 identifies the three items related to student motivation that show significant positive change (p < 0.05) between the beginning and end of the course. It is important to note that some students who took the pre-course survey did not take the post-course survey, as they did not complete the course.

Table 6. Likert Survey Items Showing Positive Impact on Motivation

|  |  |  |
| --- | --- | --- |
| Item ID | Description | Significance |
| H7 | I enjoyed working on an HFOSS project because it allowed me participate in a diverse community of HFOSS developers. | t(99) = 10.13 |
| H8 | Working on an HFOSS project increases my interest in computing. | t(102) = 5.53 |
| H10 | Working on an HFOSS project increased my confidence in my computing ability. | t(103) = 4.34 |

H7 indicates that students are motivated by working within a diverse HFOSS community. It would be hoped that this experience would result in an increase in professional skills and increase in desire to improve such skills.

H10 indicates that students perceive an increase in confidence in computing ability from working in an HFOSS project. This result, while not unexpected, could indicate that students gained proficiency in operating within an HFOSS project. This proficiency could lead to the increase in confidence.

There are also a set of software engineering learning items that show significant (p < 0.05) positive movement from the beginning of the course to the end. Table 7 shows these items.

Table 7. Likert Survey Items Showing Positive Impact on Software Engineering Learning

|  |  |  |
| --- | --- | --- |
| Item ID | Description | Significance |
| SE1 | I can list the high-level phases that comprise a software project in a real-world environment. | t(95) = 4.73 |
| SE2 | I am comfortable that I could participate in the planning and development of a real-world software project. | t(104) = 4.17 |
| SE3 | I can list the steps in the software process we used in HFOSS project. | t(96) = 12.53 |
| SE4 | I can use a software process to develop an HFOSS project. | t(99) = 8.63 |
| SE6 | I have gained some confidence in collaborating with professionals from a variety of locations and cultures. | t(104) = 6.28 |
| SE9 | I can identify the steps to be taken in maintaining an HFOSS project. | t(96) = 4.88 |
| SE11 | I can describe the drawbacks and benefits of FOSS to society. | t(104) = 9.88 |
| SE12 | I can describe the drawbacks and benefits of FOSS to business. | t(99) = 5.1 |

The perceived student learning from the items in Table 7 fall into three general categories. The first category is learning related to software engineering development and process. Items SE1 through SE4, and SE9 indicate that students are gaining an understanding the steps and process needed to develop software. Students also appear to be gaining a comprehensive understanding of software engineering as they indicate an increase in learning related to both the entire software process as well as the steps required to maintain the project. This demonstrates that students are gaining a high-level understanding of the discipline.

Item SE6 speaks directly to students’ perception that they have gained professional skills. It could also indicate that students are gaining desired skills in distributed software development. In order to collaborate with professionals in an HFOSS project, students need to understand how to communicate with developers from a variety of cultures who work in a variety of time-zones. They need the ability to clearly and concisely convey their ideas, to understand the ideas and suggestions made by others, and to be able to adapt solutions proposed by others to their own environment.

Students also indicated that they gained an understanding of the benefits and drawbacks of FOSS to society and to business. It is logical that students would gain this understanding from working in an HFOSS project and it is reassuring that they report that they’ve gained a better understanding from their experiences.

In addition to the significant differences from pre- to post-course survey, there were two items that showed close to a significant difference. Table 8 shows these items.

Table 8. Likert Survey Items Showing Moderately Positive Impact

|  |  |  |
| --- | --- | --- |
| Item ID | Description | Significance |
| H2 | I have a greater awareness of the potential for computing to benefit society due to working on an HFOSS project. | t(106) = 3.061,  P = .083 |
| O4 | Participation in an HFOSS project has caused me to consider taking further computing courses. | t(92) = 3.69,  P = .058 |

Item H2 indicates that students gained some understanding of the “h” portion of HFOSS. While significant to only P < 0.1, these results hint that students are gaining an understanding of how computing may be used to benefit society. Item O4 appears to indicate that participation in HFOSS provided incentive for students to take more computing courses. A large number of students in the study were seniors and would likely respond to this item with either “Disagree” or “Not Applicable” as they were graduating and not planning on taking additional classes. The result observed in Table 8 could be stronger if the seniors were eliminated from the study of this item.

The impact of gender and institution were also investigated but no significant results were found. The number of women was small enough that measuring any significant result was unlikely.

**Observations and Future Directions**

Based upon the results described in section 5 student participation in HFOSS does have a positive impact on both perceived student motivation and perceived student learning. Students believe that they are gaining an understanding of how to develop software in a professional, distributed environment by participating in an HFOSS project. In addition, students appear to be motivated by participating in an HFOSS project.

Hypothesis ***H1a****: Student participation in an HFOSS project has a positive impact on student motivation to study computing* has been shown to be true. Students indicate that they enjoyed working in a diverse community of developers and that working on an HFOSS project increased interest in computing. Another benefit was students’ perceived increase in confidence in their development skills. All of these factors point towards students having increased motivation to study computing.

There is strong evidence to support hypothesis ***H2a****: Student participation in an HFOSS project has positive impact on perceived learning related to software engineering*. Results indicate that students perceive that they are learning about software engineering development and process. Results also appear to indicate that they are gaining experience in developing software in a distributed environment with all of the attendant skills such as communication, distributed teamwork and more.

The study found only weak evidence to support hypothesis ***H3a****: Student participation in an HFOSS project has positive impact on major and career plans*. This could be due to the phrasing of some of the items. For instance, item O1 states: “Participation in an HFOSS project has caused me to consider computing as a major or minor.” However most of the students surveyed were already computer science majors and therefore would not necessarily agree that participation in HFOSS caused them to consider computing as a major. Therefore, questions about impact on major and career plans need to be revised in future studies.

While the overall results appear to indicate that students gain professional experience and are motivated by participating in an HFOSS project, there are some confounding variables to this study. The actual classes taught varied widely as did the location. The HFOSS projects in which students participated varied. The size and complexity of HFOSS projects can vary and smaller, simpler projects could be easier for students to work with, perhaps resulting in different impact on motivation and learning. Some HFOSS projects have a more supportive community than others which could also impact student perception of learning and motivation.

These initial promising results lead to a set of other areas to be explored. First, the ability to match pre- and post-surveys for individual students would provide a clearer picture of student perceptions of their learning and motivation. Second, it would be useful to know what aspect exactly of student involvement in HFOSS is motivating. Is it the interaction with professionals? The real-world project? The ability to make a real contribution? Third it would be interesting to know what exactly the students are learning. Are they learning communication skills? Tools? Technology? Lastly, it would be very interesting to know how students could be better scaffolded during their involvement in an HFOSS project. How could student learning be improved?

Student participation in HFOSS projects holds much promise in higher education. In addition to gaining real-world experience and helping others, students are motivated by their work. Although it is not known if participation in HFOSS can be a draw to students who are not computing students, the authors suspect the altruistic nature of HFOSS can debunk negative stereotypes about computing in general and draw students from under-represented groups.

**APPENDIX**

In this appendix, we provide the complete post survey given to students in this study. The most recent version of the survey can be found at <http://foss2serve.org/index.php/Evaluation_Instruments>.

**Student Survey - Post  
Humanitarian Free and Open Source Software**

Version 1.0

**Introduction**

This survey is part of a research project to study use of humanitarian free and open source software projects in computing courses. Your participation in this research is optional.

Information from the questionnaire will be used for statistical studies and not in any form that identifies individuals. Your answers on this questionnaire have no bearing on your grade in any course.

**Instructions**

Please answer the questions in the sections that follow. You may skip any questions that you do not want to answer for any reason. If you have any questions about this study, contact [Institutional contacts].

Thank you.

**Background**

What is your age? What is your gender?

|  |  |  |
| --- | --- | --- |
| 🞎 | 18 – 20 | 🞎 Male |
| 🞎 | 21- 23 | 🞎 Female |
| 🞎 | 23 or older |  |

What is your major?

First major:

Second major:

|  |  |
| --- | --- |
| 🞎 | Undeclared |

Do you plan to continue in this major?

|  |  |  |
| --- | --- | --- |
| Yes | No | Maybe |
| 🞎 | 🞎 | 🞎 |

If you are thinking about changing majors, why?

On a scale from 1 (one) to 5 (five), where one is beginner and five is advanced, how would you characterize your programming ability?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Beginner  1 | 2 | 3 | 4 | Advanced  5 |
| 🞎 | 🞎 | 🞎 | 🞎 | 🞎 |

Please circle the answer on the 7-point scale that indicates your agreement or disagreement with each statement.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | H-FOSS Statements | strongly disagree | dis-  agree | neutral | agree | strongly agree | not applic-able | don't know |
| H1 | Working on an H-FOSS project gives me a better appreciation for the usefulness of computing. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H2 | I have a greater awareness of the potential for computing to benefit society due to working on an H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H3 | I wanted to work on an H-FOSS project because I want to help the people who would benefit from the software. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H4 | Participating in an H-FOSS project inspires me to use my computing skills to help others. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H5 | Knowing that my project will help people motivates me to do my best on the H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H6 | Working with an H-FOSS community to develop a project has increased my interest in computing. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H7 | I enjoyed working on an H-FOSS project because it allowed me participate in a diverse community of H-FOSS developers. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H8 | Working on an H-FOSS project increases my interest in computing. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H9 | I enjoyed working on an H-FOSS project because the project can positively impact the world. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H  10 | Working on an H-FOSS project increased my confidence in my computing ability. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| H  11 | Participating in an H-FOSS project made me more comfortable with computing. | 1 | 2 | 3 | 4 | 5 | N/A | DK |

|  | Software Engineering Statements | strongly disagree | dis-agree | neutral | agree | strongly agree | not applic-able | don't know |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SE1 | I can list the high-level phases that comprise a software project in a real-world environment. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE2 | I am comfortable that I could participate in the planning and development of a real-world software project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE3 | I can list the steps in the software process we used in H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE4 | I can use a software process to develop an H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE5 | I am sure that I can actively participate in an H-FOSS community to develop a software project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE6 | I have gained some confidence in collaborating with professionals from a variety of locations and cultures. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE7 | I can describe the impact of project complexity on the approaches used to develop software. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE8 | I can describe the impact of project size on the approaches used to develop software. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE9 | I can identify the steps to be taken in maintaining an H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE10 | I am confident that I can maintain an H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE11 | I can describe the drawbacks and benefits of OSS to society. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE12 | I can describe the drawbacks and benefits of OSS to business. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE13 | I can use all tools and techniques employed in my H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE14 | I can participate in an H-FOSS development team’s interactions. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE15 | I can identify when peers in an H-FOSS project are behaving in an unprofessional manner. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| SE16 | Participation in an H-FOSS project has improved my understanding of how to behave like a computing professional. | 1 | 2 | 3 | 4 | 5 | N/A | DK |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Other Statements | strongly disagree | dis-agree | neutral | agree | strongly agree | not applic-able | don't know |
| O1 | Participation in an H-FOSS project has caused me to consider computing as a major or minor. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| O2 | Participation in an H-FOSS project has positively reinforced my decision to make computing my major. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| O3 | Participating in an H-FOSS project has caused me to question my decision to make computing my major. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| O4 | Participation in an H-FOSS project has caused me to consider taking further computing courses. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| O5 | The subject matter of this H-FOSS project is highly relevant to my future career plans. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| O6 | I have a high level of experience in the H-FOSS subject matter. | 1 | 2 | 3 | 4 | 5 | N/A | DK |
| O7 | Overall, I am very satisfied with my learning in the H-FOSS project. | 1 | 2 | 3 | 4 | 5 | N/A | DK |

**ACKNOWLEDGMENTS**

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

**REFERENCES**

Auer, Y. Juntunen, J. and Ojala, P. 2011. “Open source project as a pedagogical tool in higher education,” *15th International Academic MindTrek Conference on Envisioning Future Media Environments (MindTrek ’11*) .New York, New York, USA: ACM Press, Sept. 2011, p. 207-213.

Beyer, S., Rynes, K. and Haller, S. 2004. Deterrents to women taking computer science courses, *IEEE Technology and Society Magazine*, Vol. 23, No. 1, pp. 21-28, Spring 2004.

Bloomfield, A., Sherriff, M., and Williams, K. 2014. A service learning practicum capstone. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14).* ACM, New York, NY, USA, 265-270. DOI=10.1145/2538862.2538974 http://doi.acm.org/10.1145/2538862.2538974

Brown, J.S., Collins, A. and Duguid, P., 1989. Situated cognition and the culture of learning, *Educational Researcher*, Vol. 18, No. 1, pp. 32-42, 1989.

Buchta, J. Petrenko, M. Poshyvanyk, D. and Rajlich, V. 2006. Teaching Evolution of Open-Source Projects in Software Engineering Courses, in *22nd IEEE International Conference on Software Maintenance*. IEEE, Sept 2006, pp. 136–144.

Carrington, D., and Kim, S. K., 2003. Teaching software design with open source software, *33rd Annual ASEE/IEEE Frontiers in Education Conference*, pp. 9-14, 2003.

Carter, L. 2006. Why students with an apparent aptitude for computer science don't choose to major in computer science, SIGCSE Bulletin, Vol. 38, No. 1, pp. 27-31, March 2006.

Chacon, S. and Borenstein, J., 2014. Modernizing CS Education with Open Source, OSCON 2014, Portland Oregon.

Cicirello, V. 2013. Experiences with a real projects for real clients course on software engineering at a liberal arts institution. *J. Comput. Sci. Coll*. 28, 6 (June 2013), 50-56.

Chavez, C.V.F.G, Terceiro, A. Meirelles, P. Santos J., C. and Kon, F. 2011. Free/Libre/Open Source Software Development in Software Engineering Education: Opportunities and Experiences. In *Fórum de Educação em Engenharia de Software (CBSoft 2011-SBES-FEES)*, 2011, São Paulo. CBSoft 2011-SBES-FEES, 2011.

Cohoon, J.M. 2002. Recruiting and Retaining Women in Undergraduate Computing Majors, *SIGCSE Bulletin*, Vol. 34, No. 2, pp. 48-52, 2002.

Collins, A. 2006. “Cognitive Apprenticeship, Chapter in Cambridge Handbook of the Learning Sciences, Series: Cambridge Handbooks in Psychology, 47-61.

Carter, L. 2011. Ideas for adding soft skills education to service learning and capstone courses for computer science students. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11). ACM, New York, NY, USA, 517-522. DOI=10.1145/1953163.1953312 http://doi.acm.org/10.1145/1953163.1953312

Costa-Soria, C. and P´erez, J. 2009. Teaching software architectures and aspect-oriented software development using open-source projects, *ACM SIGCSE Bulletin*, vol. 41, no. 3, p. 385, Aug 2009.

Ding, H. 2007. Open Source: Platform for virtual service learning and user-initiated research. *Professional Communication Conference*, 2007. IPCC 2007. IEEE International , vol., no., pp.1,5, 1-3 Oct. 2007 doi: 10.1109/IPCC.2007.4464080

Ellis, H.J.C., Hislop, G.W., Chua, M., and Dziallas, S., 2011. How to Involve Students in FOSS Projects, The 2011 Frontiers in Education Conference, T1H-1,T1H-6 .

Ellis, H.J.C., Hislop, G.W., Rodriguez, J., and Morelli, R.A. 2012. “Student Software Engineering Learning via Participation in Humanitarian FOSS Projects,” 119th Annual ASEE Conference and Exhibition, Austin, TX.

Ellis, H.J.C., Purcell, M., and Hislop, G. 2012. An Approach for Evaluating FOSS Projects for Student Participation, *SIGCSE 2012, Technical Symposium on Computer Science Education*, Raleigh, NC, Mar. 2012.

Exter, M. 2014. Comparing educational experiences and on-the-job needs of educational software designers. In *Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14)*. ACM, New York, NY, USA, 355-360. DOI=10.1145/2538862.2538970 <http://doi.acm.org/10.1145/2538862.2538970>

Gehringer, E.F. 2001. From the manager’s perspective: Classroom contributions to open-source projects, in *Frontiers in Education Conference (FIE)*, IEEE, Oct 2011, pp. F1E–1–F1E–5

Glott, R., Andreas, M., Sulayman, K., et al. 2007. Report to FLOSSCom - Using the Principles of Informal Learning Environments of FLOSS Communities to Improve ICT Supported Formal Education: Phase 1 - Analysis of the Informal Learning Environment of FLOSS (Free/Libre Open Source Software) Communities, [http://www.scribd.com/doc/1949795/Report-on-the-learning-environment-of-Free-Libre-Open-Source-Software-FLOSS-communities Retrieved 1/8/12](http://www.scribd.com/doc/1949795/Report-on-the-learning-environment-of-Free-Libre-Open-Source-Software-FLOSS-communities%20Retrieved%201/8/12).

Goldweber, M., Davoli, R., Currie Little, J., Riedesel, C., Walker, H., Cross, G., and. Von Konsky, B.R. 2011. Enhancing the social issues components in our computing curriculum: computing for the social good. *ACM Inroads* 2, 1 (February 2011), 64-82. DOI=10.1145/1929887.1929907 http://doi.acm.org/10.1145/1929887.1929907

Gokhale, S.S., Smith, T., and McCartney, R. 2012. Integrating Open Source Software into software engineering curriculum: Challenges in selecting projects, in *1st International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)*, IEEE, pp.9–12.

Hislop, G.W., Ellis, H.J.C., and Morelli, R.A., “Evaluating Student Experiences in Developing Software for Humanity,” *The 14th Annual Conference on Innovation and Technology in Computer Science Education,* Paris, Jul. 2009.

Homkes, R. 2008. Assessing it service-learning. In *Proceedings of the 9th ACM SIGITE conference on Information technology education (SIGITE '08).* ACM, New York, NY, USA, 17-22. DOI=10.1145/1414558.1414564 <http://doi.acm.org/10.1145/1414558.1414564>

Jaccheri, L. and Osterlie, T. 2007. Open Source Software: A Source of Possibilities for Software Engineering Education and Empirical Software Engineering. in *1st Intl. Workshop on Emerging Trends in FLOSS Research and Development (ICSE Workshops 2007)*. IEEE, 2007, pp.5–5.

Jacobs, S. 2010. Building an education ecology on serious game design and development for the One Laptop Per Child and Sugar platforms: A service learning course builds a base for peer mentoring, cooperative education internships and sponsored research, *Games Innovations Conference (ICE-GIC), 2010 International IEEE Consumer Electronics Society's* , vol., no., pp.1,6, 21-23 Dec. 2010, doi: 10.1109/ICEGIC.2010.5716882

Kussmaul, C. 2009. Software Projects Using Free and Open Source Software: Opportunities, Challenges, and Lessons Learned. In *ASEE Annual Conference and Exposition*, 2009

Kussmaul, C., Ellis, H.J.C., and Hislop, G.W., 2012. 50 Ways to be a FOSSer: Simple Ways to Involve Students & Faculty.  *Proceedings of the 43rd SIGCSE technical symposium on Computer Science Education,* Raleigh, NC.

Lave, J. and Wenger, E., 1991. Situated learning: Legitimate peripheral participation.

Cambridge, Cambridge University Press. 1991.

Liu, C. 2005. Enriching software engineering courses with service-learning projects and the open-source approach, in *27th International Conference on Software Engineering (ICSE 2005)*. IEEE, 2005, pp. 613–614.

Ludi, S. 2011. The benefits and challenges of using educational game projects in an undergraduate software engineering course, Proceedings of the 1st International Workshop on Games and Software Engineering, 13-16.

Lundell B. Persson A. and Lings B. 2007. Learning Through Practical Involvement in the OSS Ecosystem: Experiences from a Masters Assignment in *Open Source Development, Adoption and Innovation* IFIP — The International Federation for Information Processing Volume 234, 2007, pp 289-294

MacKellar, B.K., Sabin, M. and Tucker, A. 2013. Scaling a framework for client-driven open source software projects: a report from three schools. *J. Comput. Sci. Coll*. 28, 6 (June 2013), 140-147.

Marmorstein, R. 2011. Open Source Contribution as an Effective Software Engineering Class Project, *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, ITiCSE ’11*, 268-272.

Martınez, J. J. M. 2009. Learning Free Software Development from Real-World Experience, in *International Conference on Intelligent Networking and Collaborative Systems*. IEEE, nov 2009, pp. 417–420

McCartney, R. Gokhale, S.S. and Smith T.M., 2012. Evaluating an early software engineering course with projects and tools from open source software, in *9th Annual Intl Conference on International Computing Education Research - ICER ’12*. ACM Press, sep 2012, p. 5.

Meiszner, A., Glott, R., and Sowe, S.K. 2008. Free / Libre Open Source Software (FLOSS) Communities as an Example of Successful Open Participatory Learning Ecosystems , *UPGRADE IX* (3) 62-68.

Morelli, R., de Lanerolle, T, and Tucker, A. 2012. The Humanitarian Free and Open-Source Software Project: Engaging Students in Service-Learning through Building Software, in *Service-Learning in the Computer and Information Sciences*, John Wiley & Sons, Inc., pp. 117-136, doi = 10.1002/9781118319130.ch5

Morelli, R. and de Lanerolle, T. 2009. FOSS 101: Engaging Introductory Students in the Open Source Movement. *Proceedings of the 40th SIGCSE technical symposium on Computer Science Education*, pp. 311-315.

Morgan, B. and Jensen, C., 2014. Lessons Learned from Teaching Open Source Software Development, *Open Source Software: Mobile Open Source Technologies*, Volume 427, p.133-142.

Nascimento, D.M.; Cox, K.; Almeida, T.; Sampaio, W.; Almeida Bittencourt, R.; Souza, R.; Chavez, C., 2013. Using Open Source Projects in software engineering education: A systematic mapping study, *IEEE Frontiers in Education Conference, 2013* pp.1837,1843, 23-26 Oct. 2013 doi: 10.1109/FIE.2013.6685155

Nandigam, J. Gudivada, V.N. and Hamou-Lhadj, A. 2008. Learning software engineering principles using open source software, in *38th Annual Frontiers in Education Conference (FIE)*. IEEE, 2008, pp. S3H–18–S3H–23.

Noyes, K. 2011. Open Source Software is Now a Norm in Businesses. *PC World,*  May 18, 2011 <http://www.pcworld.com/article/228136/open_source_software_now_a_norm_in_businesses.html>

Petrenko, M. Poshyvanyk, D. Rajlich, V. Buchta, J. 2007. Teaching Software Evolution in Open Source, *Computer*, vol. 40, no. 11, pp.25–31, nov 2007

Sabin, M. 2011. Free and open source software development of IT systems. In *Proceedings of the 2011 conference on information technology education* (SIGITE '11). ACM, New York, NY, USA, 27-32. DOI=10.1145/2047594.2047601 http://doi.acm.org/10.1145/2047594.2047601

Santore, J., Lorenzen, T., Creed, R., Murphy, D. and Orcutt, R. 2010. The software engineering class builds a GUI for subversion, ACM SIGCSE Bulletin, vol. 41, no. 4, p. 82, Jan 2010

Smith, T.M., McCartney, S., Gokhale, S.S. and Kaczmarczyk, L.C. 2014. Selecting open source software projects to teach software engineering. *In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14).* ACM, New York, NY, USA, 397-402. DOI=10.1145/2538862.2538932 <http://doi.acm.org/10.1145/2538862.2538932>

Stone, J.A. and Madigan, E. 2011. Experiences with community-based projects for computing majors. *J. Comput. Sci. Coll.* 26, 6 (June 2011), 64-70.

Tillberg, H. K., and Cohoon, J. M. 2005. Attaching Women to the CS Major, *Frontiers: A Journal of Women Studies*, Vol. 26, No. 1, pp. 126-140, 2005.

Toth, K. 2006. Experiences with Open Source Software Engineering Tools, *IEEE Software*, vol. 23, no. 6, pp. 44–52, nov 2006

Venkatagiri, S. 2006. Engineering the software requirements of nonprofits: a service-learning approach. In *Proceedings of the 28th international conference on Software engineering* (ICSE '06). ACM, New York, NY, USA, 643-648. DOI=10.1145/1134285.1134382 http://doi.acm.org/10.1145/1134285.1134382

Weller, M., Meizsner, A., Sowe, S.K. and Karoulis, 2008. “A Report to FLOSSCom - Using the Principles of Informal Learning Environments of FLOSS Communities to Improve ICT Supported Formal Education: Phase 2 - Report on the effectiveness of a FLOSS-like learning community in formal educational settings, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.4225&rep=rep1&type=pdf>

Wenger, E., 1999. Communities of practice: Learning, meaning, and identity, Cambridge

University Press, 1999.

Xing, G. 2010. Teaching software engineering using open source software, in *48th Annual Southeast Regional Conference on* - (ACM SE ’10) .New York, New York, USA: ACM Press, apr 2010, p. 1

Ye, Y. and Kishida, K., 2003. Toward an Understanding of the Motivation of Open Source

Software Developers, *25th International Conference on Software Engineering* , pp.

419-429, 2003.

Received August 2014; revised yyyy; accepted yyyy

1. [↑](#footnote-ref-1)