AC 2012-5220: STUDENT SOFTWARE ENGINEERING LEARNING VIA PARTICIPATION IN HUMANITARIAN FOSS PROJECTS

Dr. Heidi J.C. Ellis, Western New England University

Heidi Ellis is Chair and Associate Professor in the Computer Science and Information Technology Department at Western New England College. She has a long-time interest in software engineering education and most recently has received NSF funding to investigate the use of humanitarian free and open source software to educate computing students. She is also currently participating in an NIH grant for developing database-driven software for biological NMR analysis.

Dr. Gregory W. Hislop, Drexel University

Gregory Hislop is a professor of information science and technology and computer science at Drexel University. His interests include software engineering, computing education, and use of technology in education. Prior to joining Drexel, Hislop spent almost 20 years in IT practice with particular emphasis on products and services for enterprise systems management.

Mrs. Josephine Sears Rodriguez, Western New England University

Josephine Rodriguez is currently the Director of the Math Center at Western New England University. Additionally, for the past four years, she has also been the Associate Coordinator of Assessment at Western New England. Previously, she taught mathematics at Trinity College in Hartford, Conn.

Prof. Ralph Morelli, Trinity College

Ralph Morelli has been teaching computer science at Trinity College since 1985. His areas of expertise include artificial intelligence, free and open source software, and computing education. He is one of the directors of the Humanitarian Free and Open Source Software (HFOSS) project, which seeks to get undergraduates engaged in computing through building free software that serves the public good. Most recently, he has become involved with mobile computing and is using App Inventor for Android to teach a course on ”Computing with Mobile Phones” as one of the pilot courses of the CS Principles project, an NSF-funded effort by the College Board to develop a new AP exam in computer science.

Student Software Engineering Learning via Participation in Humanitarian FOSS Projects

Abstract

Software engineering education has long sought to provide students with real-world software development and professional experience. The use of Free and Open Source Software (FOSS) projects is one attractive approach for providing students with easy access to a complex, on-going project of size that is supported by a professional community. Humanitarian FOSS (HFOSS) projects hold the additional appeal to students of developing software that will benefit the human condition. However, student involvement in HFOSS projects can be somewhat unpredictable and less controllable than the development of home-grown projects or projects with an industry partner. Student participation in an HFOSS project means that students are dependent, at least somewhat, on the goals, schedule, and constraints of the HFOSS project itself. Therefore, learning is somewhat reliant on the progress of the HFOSS project. This paper presents results of a multi-year study of student perceptions of learning related to software engineering knowledge and skills while involved in an HFOSS project. The paper includes a background of work in student participation in HFOSS, an outline of the study approach and an explanation of the results. Implications of the results and future directions are also described.

1. Introduction

Software engineering education has a broad emphasis on students gaining experience with a real-world project and on obtaining an understanding of professional practice including such skills as teamwork, communication, work ethic, self confidence and more. The SE 2004 curriculum guidelines state "The education of all software engineering students must include student experiences with the professional practice of software engineering."[1, (pg 9)] Indeed, the Computer Science CC 2005 guidelines[2] suggest that students gain both technical knowledge and professional skills via participation in a real-world project. Part of the program criteria for Software Engineering programs for ABET[3] includes the need for students to work in at least one significant application domain. In addition, Begel and Simon[4] identified that professional issues such as communication and collaboration skills are key to the success of new graduates. One main mechanism for providing students with professional experience and skills in developing complex software systems is involvement in a real-world project within the classroom environment or internship.

There are several different models for student involvement in software projects. One less-commonly used model is students developing projects within a classroom where the project is defined by the instructor, the students, or with some combination of instructor and student input[5,6,7]. This approach has the benefits of the project being entirely under the control of the instructor and the experience can be tailored to meet specific learning objectives. The downsides include lack of a professional development community to help build professional skills and the possibility that the code may not be used. These projects are frequently Greenfield projects which do not provide exposure to the complexities of existing code. In addition, student motivation can be an issue with this type of project as many of these projects may not actually be used by a customer. In one ideal situation, the students themselves are customers of the end project and are therefore vested in the outcome[7]. Games also provide a significant motivation for these types of projects[5].

Another model of student projects is student teams working on a homegrown project where a faculty member or local academic representative, such as the institution's admissions office or local non-profit, serve as customer[8,9,10]. This approach allows the instructor some control over the course while also providing students with the motivation of a "real" customer. As with the classroom project model, these projects also suffer from the lack of a professional community, although an outside project provides experience with customer interactions. In addition, the possibility of code being used and perhaps sold means that the issue of intellectual property will likely need to be resolved if the students have any desire of creating a commercial product.

Perhaps the most popular model for student projects is having students collaborate with an outside industry partner who proposes a project and acts as the customer for that project[11-14]. Many of these projects involve making an enhancement or modification to an existing project of size and are very successful at teaching students software engineering skills. These projects provide students with a professional community with which to interact. In addition, in some cases the development may be distributed, providing students with experience in international software development. The disadvantages of these projects include the lack of transparency of artifacts and issues with intellectual property. Students typically are restricted from showing artifacts from this type of project to potential employers as evidence of their accomplishments.

Student projects based in FOSS are a model of projects that is growing[15-24]. In the FOSS project model, ideally students become part of a FOSS community and contributions that meet course learning objectives are identified. The FOSS community is the customer and students work within the community to complete a project. The benefits of this approach are that FOSS projects are inherently open, providing instructors and students with easy access to an on-going project of size. FOSS projects are also supported by a professional community, frequently international, which supplies students with desired professional experience. Intellectual property issues may be simplified by the open source licensing used in the project. In addition, contributions to FOSS projects provide students with easily accessible evidence of their professional capabilities, and the FOSS community may even provide student recommendations for employment.

The disadvantages to the FOSS-based approach include that many instructors are not familiar with FOSS tools and techniques which results in a learning curve for the instructor. In addition, faculty members must gain an understanding of the FOSS community interactions. The volunteer-driven nature of FOSS projects also can present problems in that such projects may have an inconsistent pattern of effort which may affect student efforts. Lastly, the release schedule of the FOSS project may be difficult to map to the academic terms, making scheduling of deliverables difficult.

2. Prior Research in Student Involvement in HFOSS

Humanitarian FOSS (HFOSS) can be broadly defined as any FOSS project that benefits the human condition in some fashion. Therefore, HFOSS projects can range from disaster relief to medical records to microfinance to educational support. Investigation by the authors into student involvement in HFOSS projects began in early 2006. Between 2007-2009, faculty members at several small, liberal arts institutions began involving students in HFOSS projects as part of the Humanitarian FOSS project (http://hfoss.org) which obtained NSF CPATH funding. The courses were software development courses, typically aimed at upper-level students, and summer internships. Results from these early experiences have shown that HFOSS projects have the ability to attract students due to their altruistic goals and the possibility for the project to benefit the human condition[25-28].

These early investigations into involving students in HFOSS projects resulted in the identification of possible roadblocks to student participation in FOSS projects including range of student backgrounds, limited course duration, uncontrolled development processes, and complexity of the project[29,30]. In addition, it was noted that the identification of software development processes that support student participation in HFOSS as well as course materials are needed to aid instructors[31]. Morelli and deLanerolle[32] demonstrated that students can actively participate in HFOSS projects at an introductory level, in addition to more advanced software engineering courses. More recently, Morelli et al[33], report on the ability of students to make an impact on the real world through the development of a version of the Sahana disaster relief software that was used in Sichuan Province, China in response to the May 12, 2008 earthquake.

More recent investigation has shed some light on providing guidance to faculty members desiring to involve students in HFOSS projects. Based on the lessons learned with initial efforts in student participation in HFOSS, Ellis et al[34], have outlined an approach to the problem of project selection for student participation in HFOSS. The approach provides criteria for evaluating potential HFOSS projects along the axes of project viability, community approachability, and suitability for use within a particular course. Ellis et al[35], have also provided guidelines to faculty members who desire to involve students in FOSS projects. The guidance includes information on understanding FOSS culture, comprehending community involvement, fostering student participation, synchronizing deliverables/assignments, and gracefully closing out involvement at the end of a term.

This paper presents the results of a multi-year survey-based study of student opinion on the software engineering aspects of learning when involved in an HFOSS project. Initial results of software engineering learning were published in 2009[36]. This paper provides a more detailed view of the results and incorporates a larger data set than the earlier effort.

3. The Study

The study discussed in this paper is a subset of a larger study into the impact of student participation in HFOSS projects. The larger study covers three main aspects of the impact of student participation:

1. The impact of participation in an HFOSS project on student attitude towards computing
2. The degree of perceived learning related to software engineering knowledge and skills
3. The impact of participation in an HFOSS project on major selection and career plans

This paper presents results of the second aspect only, focusing on the software engineering learning that students perceived.

The research question investigated in the study is whether participation in HFOSS projects impacts the perception of student learning in the area of software engineering:

> **H_o**: Student involvement in an HFOSS project has no impact on perceived learning of software engineering knowledge
> **H_a**: Student involvement in an HFOSS project has a positive impact on perceived learning of software engineering knowledge

The study presented in this paper involved ten courses offered at four different small, liberal arts academic institutions between summer 2008 and fall term 2010. The courses were 15-week semesters taught in either the fall or spring semester and ranged from software engineering and software development courses aimed at juniors and seniors to an introductory computing course. Class sizes were small, with typically no more than 20 students per class. The study also encompassed three 10-week summer internships. These internships were more focused than the courses, where students concentrated on making specific enhancements to HFOSS projects. Across both courses and internships, students participated in a range of HFOSS projects ranging from disaster management applications to applications to aid disabled computer users.

The main study instrument was a five-point Likert scale survey with response values "strongly disagree", “disagree”, “neutral”, “agree”, and "strongly agree". “Don’t know” and “Not applicable” options were also provided. Survey items were developed in consultation with a cultural anthropologist based in the goals of the study. The survey items addressed both understanding of software engineering tools and approaches, as well as professional skills gained. When possible, a pre-course and post-course version of the survey was administered. In the case where both pre- and post-course surveys were administered, the pre-course surveys were not matched with the corresponding post-course survey response. Table 1 below shows the questions that were contained in the software engineering portion of the survey.

SE1	I can list the high-level phases that comprise a software project in a real-world environment.
SE2	I am comfortable that I could participate in the planning and development of a real-world software project.
SE3	I can list the steps in the software process we used in HFOSS project.
SE4	I can use a software process to develop an HFOSS project.
SE5	I am sure that I can actively participate in an HFOSS community to develop a software project.
SE6	I have gained some confidence in collaborating with professionals from a variety of locations and cultures.
SE7	I can describe the impact of project complexity on the approaches used to develop software.
SE8	I can describe the impact of project size on the approaches used to develop software.
SE9	I can identify the steps to be taken in maintaining an HFOSS project.
SE10	I am confident that I can maintain an HFOSS project.
SE11	I can describe the drawbacks and benefits of FOSS to society.
SE12	I can describe the drawbacks and benefits of FOSS to business.
SE13	I can use all tools and techniques employed in my HFOSS project.
SE14	I can participate in an HFOSS development team’s interactions.
SE15	I can identify when peers in an HFOSS project are behaving in an unprofessional manner.
SE16	Participation in an HFOSS project has improved my understanding of how to behave like a computing professional.

Table 1. Software Engineering Survey Items

The survey responses were converted to an ordinal number from one to five with one representing the “strongly disagree” response and the five representing the “strongly agree” response. This allowed observations to be made about collective student responses.

4. The Results

This section provides an analysis of the results of the survey. The section contains a discussion of the hypothesis tests for the mean as well as describing the results of investigations into comparison of pre- and post-course results, the impact of student self-assessed programming ability, and the impact of gender. The discussion starts with an overview of the data collected. Table 2 below summarizes the courses in the study and survey responses.

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	Internship	Post	15	5	33.33%
Fall 2010	Sr Course	Pre	11	11	100.00%
Fall 2010	Sr Course	Post	11	11	100.00%
		Total	**158**	**133**	**84.18%**

Table 2. Summary of Survey Participation

The response rate to the surveys was very good with an average response rate of 84.18%. This may be the result of the small class size and the fact that surveys were administered via paper form during a class or internship meeting. 87 (65.4%) of the responses were from male students and 27 (20.3%) of the responses were from female students. 19 (14.3%) responses left the gender answer blank.

Out of a total of 133 responses, 72 responses were from students between 18 and 20 years of age, 56 responses were from students between 21 and 23 years of age, four responses were from students older than 23 and there was one blank response. Table 3 below shows the distribution of majors of the responses.

Major	Responses	Percent
CS	73	54.89%
Undecided	19	14.29%
Math	8	6.02%
Comp. Eng.	6	4.51%
Blank	3	2.26%
Rest	24	18.05%

Table 3. Majors of Respondents

As the source schools were small, liberal arts institutions, it is natural that the majority of responses are from students in Computer Science programs. It was interesting to note that 14% of the students who responded to the survey were undecided about their major. The majors of the 24 students in the "Rest" category ranged from Molecular Biology to Economics to English to Political Science.

Hypothesis Test for Mean

The post-course data provides a retrospective on student experience with participating in an HFOSS project. Results from both courses and internship experiences were included in the analysis. To provide a high-level view of the results, Figure 1 below shows the average of the 79 post-course survey results. Observations of Figure 1 below show that students rated all of the software engineering items above the **neutral** rating of 3.0.

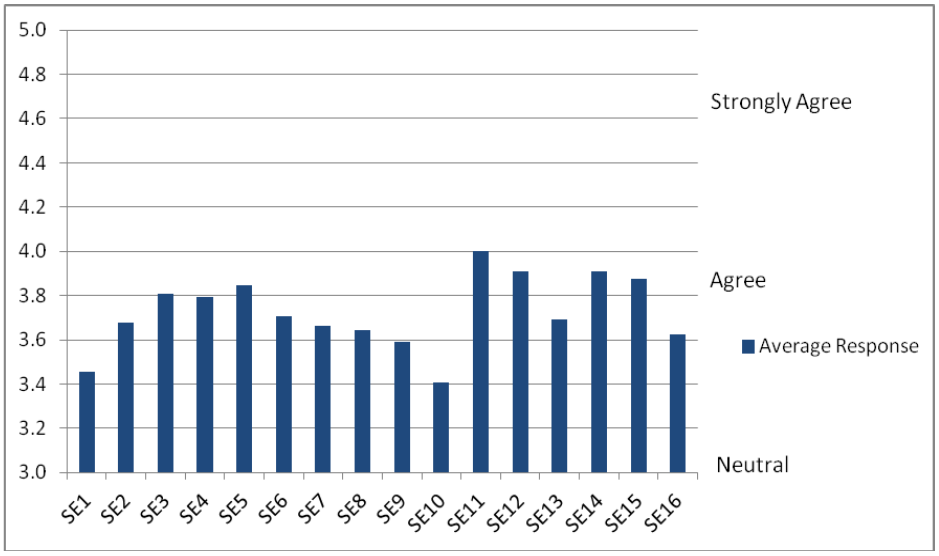

Figure 1. Mean Post-Course Survey Item Responses (All Questions Above the Neutral Rating of 3.0)

The survey items with the four highest average response are either at or very close to the **agree** level of response. Items SE 11 and SE 12, which indicate student ability to describe drawbacks and benefits of FOSS to society and to business, have average responses of 4.0 and 3.91 respectively. Clearly, students perceive that they have an understanding of FOSS's role in professional world. The 3.91 average for item SE14 indicates that students feel able to participate in the interactions of an HFOSS development team and the 3.88 rating for item SE15 appears to indicate that students have gained an understanding of developing within a professional environment and when peers are behaving in an unprofessional manner.

Upper-tailed hypothesis tests for the mean were performed[37], assuming a mean response value of the neutral rating of 3 in the null hypothesis. The results indicated that there was sufficient evidence to conclude the alternative hypothesis that student participation in HFOSS positively impacts student learning for all 16 survey items. These results provide very strong evidence that student participation in HFOSS projects positively impacts their perceived learning ($p<=0.0001$ in each case).

Pre/Post Survey Comparisons

One aspect of interest was how student opinion changed from the beginning of the course to the end. In this case, only courses were considered (not internships). As the pre- and post-course surveys for individual students could not be matched, the analysis compared the average results of the pre- and post-course surveys. The data set included 48 pre-course surveys and 51 post-course surveys. A 2-tailed t-test for the difference of means (assuming unequal variance[37]) revealed eight survey items with a significant positive difference ($p <= 0.1$), shown in Table 4 below.

ID	**Item**	***p*-value**
SE1	I can list the high-level phases that comprise a software project in a real-world environment.	0.012
SE3	I can list the steps in the software process we used in HFOSS project.	0.000
SE4	I can use a software process to develop an HFOSS project.	0.012
SE7	I can describe the impact of project complexity on the approaches used to develop software.	0.042
SE8	I can describe the impact of project size on the approaches used to develop software.	0.012
SE11	I can describe the drawbacks and benefits of FOSS to society.	0.029
SE12	I can describe the drawbacks and benefits of FOSS to business.	0.045
SE13	I can use all tools and techniques employed in my HFOSS project.	0.018

Table 4. Software Engineering Survey Items
(Demonstrating Significantly Stronger Post-Course Responses)

Observing Table 4, it is apparent that students perceive that they have gained software engineering knowledge in the areas of tools and techniques. It appears that students have gained technical knowledge about the steps and tools used to develop and HFOSS project. They also appear to have a better idea of the impact of FOSS on business and society by the end of the course. This could be a natural result of having spent a term working on an HFOSS project of some size and complexity.

Two survey items had a significant negative difference ($p <= 0.05$). It is interesting to note that these two items are related to the professional aspect of software engineering. SE6 ($p = 0.002$) addresses student confidence gained in collaborating with a range of professionals and SE16 ($p = 0.006$) addresses student improvement in understanding of how to behave like a computing professional. As responses to both items showed a significant negative response, it appears that students feel less comfortable operating with a professional environment, which may be the result of obtaining a better perspective on the magnitude of professional society.

Impact of Self-Assessed Programming Ability

Another aspect of interest was the impact of students' self-assessed programming ability on post-course survey responses. For this analysis, the post-course surveys for all courses and internship experiences were used (N=79). Students self-assessed their programming experience on a scale of one to five with one having a value of "Beginner" and five having a value of "Advanced". The values one through three were collected into to the "low" programming ability group and values four and five were collected into to the "high" programming ability group. 43 students self-assessed their ability at level one through three and 36 students self-assessed their ability at level four or five.

A 2-tailed t-test for the difference of means (assuming unequal variance) revealed nine survey items where students with higher programming ability showed significantly stronger responses (p <= 0.1), shown in Table 5 below.

ID	**Item**	***p*-value**
SE1	I can list the high-level phases that comprise a software project in a real-world environment.	0.000
SE2	I am comfortable that I could participate in the planning and development of a real-world software project.	0.000
SE3	I can list the steps in the software process we used in HFOSS project.	0.053
SE4	I can use a software process to develop an HFOSS project.	0.055
SE5	I am sure that I can actively participate in an HFOSS community to develop a software project.	0.079
SE7	I can describe the impact of project complexity on the approaches used to develop software.	0.065
SE10	I am confident that I can maintain an HFOSS project.	0.093
SE13	I can use all tools and techniques employed in my HFOSS project.	0.015
SE14	I can participate in an HFOSS development team's interactions.	0.075

Table 5. Software Engineering Survey Items
(Comparison of "Low" Programming Ability vs. "High" Programming Ability)

Since the students that self-rated with a higher programming ability are likely to have more software development experience, it is not unexpected that this group of students should show a significantly stronger response to the software engineering items in Table 5. It is interesting to note that items SE5 and SE14, which address the professional aspects of software engineering, also have strong results. This may imply that students are gaining professional skills along with technical skills, one of the desired effects of participating in an HFOSS project. No survey items showed significant negative responses, meaning that the low self-assessing students did not show a significantly stronger response than the high self-assessing students.

Impact of Gender
One aspect of particular interest was the analysis of the impact of gender on results for software engineering survey items. As with previous analyses, the post-course surveys were used

resulting in 13 responses from females and 56 responses from males. 10 survey responses indicated no gender and these results were not included in the analysis. A 2-tailed t-test for the difference of means (assuming unequal variance) revealed three survey items with a significant positive difference (p <= 0.1) as shown in Table 6 below.

ID	Item	*p*-value
SE1	I can list the high-level phases that comprise a software project in a real-world environment.	0.025
SE2	I am comfortable that I could participate in the planning and development of a real-world software project.	0.025
SE10	I am confident that I can maintain an H-FOSS project.	0.071

Table 6. Software Engineering Survey Items
(Demonstrating Significantly Stronger Responses by Females)

It is interesting to note that females appeared to indicate that they had a stronger high-level understanding of the software engineering aspects of an HFOSS project. Results show that females felt that they were better able to comprehend the parts that make up an HFOSS project and better able to plan, develop and maintain such a project. The sample size for this analysis is relatively small and a larger female group would provide stronger evidence for these results.

5. Discussion

The results of student opinion of software engineering knowledge gained while working with an HFOSS project are promising. The results of the hypothesis test of the mean analysis where all 16 survey items showed a significant positive response is a clear indication that students feel that they are gaining software engineering knowledge from working with an HFOSS project. In addition, the results underscore that students are gaining understanding of professional issues and how to operate in a professional development environment via this experience. Generally, this leads to the conclusion that significant learning takes place when working with an HFOSS project.

The investigation of the change in student opinion of software engineering learning from the beginning of the course to the end provides evidence that students feel that they are gaining facility in the tools and approaches used to develop an HFOSS project. The positive responses in this analysis appear to be focused on the technical aspects of project development. The significant decrease in two survey items related to professional communication appear to indicate that students are less comfortable in a professional environment at the end of the course than at the beginning. This may result from students obtaining a much clearer view of professional aspects of the discipline and a realization that the discipline is larger and more complex than they had originally thought. One natural result of a student working on an HFOSS project for the first time would be a broader vision of the field of software development.

The comparison of survey responses for low and high programming ability support the logical results that students that self-assess with stronger programming skills are more proficient in the technical and professional aspects of software engineering than those that self-assessed at a lower programming ability. One reason for this result could be that students with more confidence in their programming skills may have more programming or software development experience and therefore may feel more secure in their software engineering skills.

The results of the investigation of the impact of gender provide interesting insight into the skills that female students perceive that they gain by participating in an HFOSS project. The responses from the female students that were significantly stronger than their male counterparts appear to indicate that they gain a higher-level understanding of the development of an HFOSS project. These results provide discernment into the possible areas of software engineering that may be more attractive to female students. These results suggest that introducing software engineering using a top-down approach and emphasizing the planning and management aspects of a project may be more effective at attracting and retaining female students.

There were some limitations to the study including courses and internships lead by different instructors, the different length and focus of courses versus internships, and the differing formats (online versus face-to-face) in which courses were offered. Additional limitations include the low number of female participants and the number of participants that did not respond to the gender item. In addition, the courses had excellent response rates for the survey, but there were low response rates from two summer internships, resulting in a wide range of response rates.

Overall, the results of the study strongly support the alternative hypothesis that student involvement in an HFOSS project positively impacts student learning in the area of software engineering. Results indicate that students gain key software engineering knowledge that can be difficult to convey in a traditional classroom. Students gain an understanding of the impact of size and complexity on an HFOSS project as well as the impact of FOSS on business and professional worlds. They also gain hands-on knowledge of software process, tools, and techniques.

6. Conclusion

The results of the study presented in this paper clearly demonstrate that students gain significant software engineering knowledge via participation in an HFOSS project. Given the identified barriers to faculty incorporating HFOSS projects in the classroom, attention needs to be paid to the infrastructure and support for instructors so that more instructors will be enabled to use this form of project. One focus of current research is on identifying ways to facilitate greater faculty participation in HFOSS projects. An area for future investigation is the economic impact of student participation in HFOSS projects.

The analysis presented in this paper is part of a larger study on the impact of HFOSS on student motivation and learning. Analysis of the other aspects of humanitarian and career impact aspects is ongoing. A closer look at the impact of the humanitarian aspect of student involvement with a project will shed light on the impact of participation in HFOSS versus participation in FOSS.

Acknowledgement

This material is based on work supported by the National Science Foundation under Grants DUE-0958204, DUE-0940925, CISE- 0722137, and CISE-0930934. 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).

Bibliography

1. Software Engineering 2004 – Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. IEEE-CS/ACM. http://sites.computer.org/ccse/SE2004Volume.pdf 2004.Accessed 1/4/12.
2. Computing Curricula 2005: Computer Science. The Overview Report. IEEE-CS/ACM. 2005. http://sites.computer.org/ccse/SE2004Volume.pdf Accessed 1/4/12.
3. Association Board for Engineering and Technology, 2012. "Criteria for Accrediting Engineering Programs, Effective for Reviews During the 2012-2013 Accreditation Cycle,"http://www.abet.org/uploadedFiles/Accreditation/Accreditation_Process/Accreditation_Documents/Current/eac-criteria-2012-2013.pdf Accessed 1/4/12.
4. Begel, A. and Simon, B., "Struggles of new college graduates in their first software development job," Proceedings of the 39th SIGCSE technical symposium on Computer science education, (2008), pp. 226-230.
5. Claypool, K. and Claypool, M., "Teaching software engineering through game design," SIGCSE Bulletin, vol. 37, no. 3, (September 2005), pp. 123-127.
6. Way, T. P.}, "A company-based framework for a software engineering course," SIGCSE Bulletin, vol. 37, no.1 (February 2005), pp. 132-136.
7. Mann, S. and Smith, L., "Software engineering class eating its own tail," Proceedings of the ninth Australasian conference on Computing education - Volume 66, (2007), pp. 115-123.
8. Alzamil, Z., "Towards an effective software engineering course project," Proceedings of the 27th international conference on Software engineering, (2005), pp. 631-632.
9. Webster, L. D. and Mirielli, E. J., "Student reflections on an academic service learning experience in a computer science classroom," Proceedings of the 8th ACM SIGITE conference on Information technology education, (2007), pp. 207-212.
10. MacKellar, B. K., "A software engineering course with a large-scale project and diverse roles for students," *Journal of Computing Sciences in Colleges*, vol. 26, no. 6, (June 2011), pp. 93-100.
11. Hogan, J.M., Smith, G., and Thomas, R., "Tight spirals and industry clients: the modern SE education experience," Proceedings of the 7th Australasian conference on Computing education - Volume 42, (2005) pp. 217-222.
12. Reichlmayr, T. J., "Collaborating with industry: strategies for an undergraduate software engineering program," Proceedings of the 2006 international workshop on Summit on software engineering education, (2006), pp. 13-16.
13. Fornaro, R.J., Heil, M.R., and Tharp, A.L., "Reflections on 10 years of sponsored senior design projects: Students win-clients win!" Journal of Systems and Software, vol. 80, no. 8, (August 2007), p. 1209-1216.
14. Krogstie, B. and Bygstad, B., "Cross-Community Collaboration and Learning in Customer-Driven Software Engineering Student Projects," Proceedings of the 20th Conference on Software Engineering Education \& Training, (2007), pp. 336-343.
15. Carrington, D., and Kim, S. K., "Teaching software design with open source software," 33rd Annual ASEE/IEEE Frontiers in Education Conference, (2003), pp. 9-14.
16. Shockey, K. and Cabrera, P., "Using open source to enhance learning," Proc. of 6th ITHET, (2005), pp. 7-12.
17. Jaccheri, L. and Osterlie, T., "Open Source Software: A Source of Possibilities for Software Engineering Education and Empirical Software Engineering," Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development, (2007), 5 pages.
18. Toth, K., "Experiences with open source software engineering tools," IEEE Software, vol. 23, no. 6, (2006), pp. 44-52.
19. Krogstie, B. R., "Power through brokering: open source community participation in software engineering student projects," Proceedings of the 30th international conference on Software engineering, (2008), pp. 791-800.
20. Nandigam, J., Gudivada, V.N., and Hamou-Lhadj, A., "Learning software engineering principles using open source software", 38th Annual Frontiers in Education Conference, (2008), pp. S3H-18 - S3H-23.
21. Kussmaul, C., "Software projects using free and open source software," Proceedings of the American Society for Engineering Education Annual Conference, Austin, TX, (2009), pp. 41-42.
22. Ludi, S. "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, (2011) pp. 13-16.

23. Marmorstein, R., "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, (2011), pp. 268-272.
24. Stroulia, E., Bauer, K., Craig, M., Reid, K., and Wilson, G., "Teaching distributed software engineering with UCOSP: the undergraduate capstone open-source project," Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development, (2011), pp. 20-25.
25. Ellis, H.J.C., Morelli, R.A., de Lanerolle, T.R., Damon, J., and Raye, J., "Can Humanitarian Open-Source Software Development Draw New Students to CS?" SIGCSE 2007, Technical Symposium on Computer Science Education, (March 2007), pp. 551-555.
26. Ellis, H.J.C., Morelli, R.A., and de Lanerolle, T., "Holistic Software Engineering Education Based on an Open Source Project," 20th Annual Conference on Software Engineering Education and Training, (July 2007), pp. 327-335.
27. Morelli, R.A., Tucker, A.L., Danner, N., de Lanerolle, T.R., Ellis, H.J.C., Izmirli, O., Krizanc, D., and Parker, G. 2009. Revitalizing Computing Education by Building Free and Open Source Software for Humanity, Communications of the ACM , vol. 52, no. 8, (August 2009), pp. 67-75.
28. Morelli, R.A., Tucker, A.L., Danner, N., de Lanerolle, T.R., Ellis, H.J.C., Izmirli, O., Krizanc, D., and Parker, G., "Revitalizing Computing Education by Building Free and Open Source Software for Humanity", Communications of the ACM , vol. 52, no 8, (August 2009), pp. 67-75.
29. Morelli, R.A., Ellis, H.J.C., de Lanerolle, T., Damon, J., and Walti, C., "Can Student-Written Software Help Sustain Humanitarian FOSS?", The 4th International Conference on Information Systems for Crisis Response and Management, Delft, the Netherlands, (May 2007), pp. 41-44.
30. Ellis, H.J.C., Morelli, R.A., and Hislop G. W., "WIP: Challenges to Educating Students within the Community of Open Source Software for Humanity," The 2008 Frontiers in Education Conference, Saratoga Springs, NY, (October 2008), pp. S3H-7 - S3H-8.
31. Ellis, H.J.C., Morelli, R.A., and Hislop, G.W., "Support for Educating Software Engineers Through Humanitarian Open Source Projects," 21st Annual Conference on Software Engineering Education and Training, Charleston, SC, (April 2008), pp. 1-4.
32. Morelli, R. and de Lanerolle, T. FOSS 101: Engaging Introductory Students in the Open Source Movement. SIGCSE09: Proceedings of the 40th ACM technical symposium on Computer science education, Association of Computing Machinery, (March 2009), pp. 311-315.
33. Morelli, R., de Silva, C., de Lanerolle, T., Curzon, R., and Mao, X. "A global collaboration to deploy help to China" Communications of the ACM, vol. 53, no. 12, (December 2010) pp.142-149.
34. Ellis, H.J.C., Purcell, M., and Hislop, G., "An Approach for Evaluating FOSS Projects for Student Participation," SIGCSE 2012, Technical Symposium on Computer Science Education, Raleigh, NC, Mar. 2012.
35. Ellis, H.J.C., Hislop, G.W., Chua, M., and Dziallas, S., "How to Involve Students in FOSS Projects," The 2011 Frontiers in Education Conference, Rapid City SD, (October 2011) .
36. Hislop, G.W., Ellis, H.J.C., and Morelli, R.A. "Evaluating Student Experiences in Developing Software for Humanity," In Proceedings of the Fourteenth Annual ITiCSE, (July 2009) pp. 263-267.
37. McClave, J.T., Benson, P.G., and Sincich, T., Statistics for Business and Economics, 9th edition, Pearson – Prentice Hall, Upper Saddle River, NJ, 2005.