ID Paper	Title	Authors	Abstract
89	Building a community system to teach collaborative software development	A. Villarrubia ; H. Kim	This paper reports an Open Source Software (OSS) community for Computer Science students to support collaborative software development activities. We built an in-house version control system using only OSS products, which allows students to easily collaborate on development projects, while simultaneously allowing instructors to easily track students' activities. As our system provides a controlled educational environment, students can experience various aspects of software development by playing different roles. In addition, the community's code repository works as a knowledge base for student projects, and thus students can reuse the code and artifacts as examples or basic frameworks for their development.
93	An Open Source approach to enhance industry preparedness of students	Smrithi Rekha V ; V. Adinarayanan	Transitioning from college to industry can be challenging task if students are not exposed to the right content while in college. In this paper we present an Open Source based approach to Software Engineering where students get the opportunity to work on live Open Source projects thereby gaining practical exposure to various Software Engineering principles. The objective of this approach is to enhance the Industry Preparedness of students and addressing the challenges they may face as professionals. This approach is based on the result of our study of Software Engineers who have newly joined the industry. The study involved understanding the Software Engineering (SE) course content they were exposed to as students, whether the SE related course(s) were able to meet the outcomes laid by Software Engineering Education Knowledge, whether the course adequately prepared them to take on various industry responsibilities and the challenges they faced in the transition.
98	Training Software Engineers Using Open-Source Software: The Professors' Perspective	G. H. L. Pinto ; F. F. Filho ; I. Steinmacher ; M. A. Gerosa	Traditional Software Engineering (SE) courses often prioritize methodologies and concepts in small, controlled environments: naive projects used as a proof of concept instead of full-fledged real software systems. Although this strategy has clear benefits, it does not place enough care in training students to face complex, non-trivial legacy software projects. To bridge this gap, novel SE courses are leveraging the rich variety of open-source software (OSS) projects to illustrate how these methodologies and concepts are applied to existing, non-trivial software systems. To better understand the benefits, challenges, and opportunities of this transition, in this paper, we interview seven SE professors that changed their academic setting to aspire students to comprehend, maintain, and evolve OSS systems as part of their SE course. We found that there are different ways to make use of OSS projects in SE courses in terms of project choice, assessment, and learning goals. Moreover, we evidence clear benefits of this approach, including improving students' social and technical skills, and helping students enhancing their resume. Also, we observed that this strategy comes with costs: the activity demands effort and time from the professor and the barrier for one getting involved with and, therefore, placing a meaningful contribution, in an OSS community is often high.

106	Utilizing open source software in teaching practice-based software engineering courses	M. Dorodchi ; N. Dehbozorgi	Software engineering courses face the challenge of covering all the stages of analysis, development, maintenance, and support while addressing practical issues such as dealing with large codebase. Free and open source software (FOSS) and more specifically humanitarian free and open source software (HFOSS) have been used by many educators to bring many add-ons to computer science education such as innovation and motivation. In addition, FOSS/HFOSS could give a better understanding of real world projects to students. In this work, we are looking at some activities developed for teaching upper division undergraduate and graduate software engineering courses using open source software projects and analyze the impacts of using this approach on students.
115	Using Intelligent Tutors to Teach Students How APIs Are Used for Software Engineering in Practice	V. Krishnamoorthy ; B. Appasamy ; C. Scaffidi	Computer science courses typically incorporate integrated training in software engineering, which includes learning how to reuse existing code libraries in new programs. This commonly presents a need to use the libraries' application programming interfaces (APIs) during project-based learning activities. The students learn these APIs with little support from the instructor. Instead, they primarily learn from whatever code examples they find on the Web-a scattershot approach that is slow, frustrating, and sometimes fruitless. Preliminary work investigated an improved approach that provides students with intelligent API tutors. This interactive educational content walks students through code examples selectively harvested from the Web. This approach increased students' knowledge of API method names, but a key pedagogical challenge remained: Students still lacked an understanding of common problems and surprises that they might encounter when using APIs in practice. Perhaps even more crucially, the prior statistical study provided no information about how students would perceive and accept the intelligent tutor approach, when given a choice. Therefore, this paper presents an enhanced approach that augments intelligent tutors with selected information from online FAQs and online open-source code, thereby providing more explanation and context about how to use APIs in practice. A qualitative study assessed student perceptions of the system's usability as well as suggestions for improvement, revealing that 100% of students considered the approach to be useful. A key implication is that providing information from online FAQs, related articles, and open-source code could facilitate computer science students' learning how to use APIs.
134	Application of software visualization in programming teaching	Yi Ding ; Yongmin Hang ; Gang Wan ; Shuiyan He	Motivating students to comprehend software structure, behavior and complexity is often difficult because software cannot be seen or even touched, disappear into files on disks. To overcome this problem, the author has prepared an open source project, some programs developed by students themselves and two software visualization tools, and applied them in programming teaching. They are suitable for classroom use in undergraduate programming courses. The effectiveness of software visualization in programming teaching has been evaluated formally by students. The implementation of software visualization in programming teaching was judged to be successful because of the positive student feedback. This paper describes the case and experiment, the overall effectiveness, and plans for further work.

135	Teaching software maintenance with open source software: Experiences and lessons	S. Gokhale ; T. Smith ; R. McCartney	Software Engineering (SE) careers are overwhelmingly devoted to the maintenance and evolution of existing, large software systems, where the key challenge is code comprehension especially in the face of inadequate documentation and support. SE courses must thus prepare students to meet this challenge. Open Source Software (OSS) furnishes a valuable source of realistic, sizeable projects for inculcating the appreciation and skills involved in code comprehension and evolution. This paper describes experiences and lessons learnt in using OSS projects to teach an introductory, sophomore/junior-level SE course with an emphasis on comprehension, maintenance, and evolution. Students' reactions and undertakings, acquired through participant observation and homework assignments, suggest that OSS can meaningfully illustrate comprehension and evolution difficulties. Finally, it describes the characteristics of OSS projects that are conducive to highlighting maintenance challenges.
1088	Localized open source collaboration in software engineering education	K. Buffardi	Involving computer science students in open source software projects provides opportunities for them to contribute to real products with more authentic scope than typical computer science assignments. However, the environment of collaborating with external, distributed teams also poses unique challenges and may distance students from the potential for valuable, direct contact and mentorship from software professionals. In addition, while the technology industry continues to grow, smaller communities have a vested interest in growing a culture for collaboration between students and local software developers. We formed a local open source organization to collaborate on a product by combining efforts from students and professionals. This paper describes the localized free and open source software (LFOSS) organization and reports initial findings from software engineering students' involvement.
1097	Humanitarian Open Source Software in Computing Education	G. W. Hislop ; H. J. C. Ellis	Empowering students to become socially responsible professionals is a desirable result of computing education. Humanitarian Free and Open Source Software (HFOSS) projects provide an opportunity for computing educators to inspire their students to tackle global humanitarian challenges while also learning about software engineering.
1192	Using Gamification to Orient and Motivate Students to Contribute to OSS Projects	G. C. Diniz ; M. A. G. Silva ; M. A. Gerosa ; I. Steinmacher	Students can benefit from contributing to Open Source Software (OSS), since they can enrich their portfolio and learn with real world projects. However, sometimes students are demotivated to contribute due to entrance barriers. On the other hand, gamification is widely used to engage and motivate people to accomplish tasks and improve their performance. The goal of this work is to analyze the use of gamification to orient and motivate undergraduate students to overcome onboarding barriers and engage to OSS projects. To achieve this goal, we implemented four gaming elements (Quests, Points, Ranking, and Levels) in GitLab and assessed the environment by means of a study conducted with 17 students, within a real OSS project (JabRef). At the end of the study, the students evaluated their experience through a questionnaire. We found that the Quest element helped to guide participants and keep them motivated and points helped by providing feedback on students' performed tasks. We conclude that the gamified environment oriented the students in an attempt to make a contribution and that gamification can motivate and orient newcomers' to engage to OSS projects.

1193	Towards a model of faculty development for FOSS in education	H. J. C. Ellis ; M. Chua ; G. W. Hislop ; M. Purcell ; S. Dziallas	Free and Open Source Software (FOSS) has become an important segment of the computing industry and a source of innovation in software development. The open culture of FOSS projects where all project artifacts are accessible and communication is visible provides computing educators with an array of unique opportunities for student learning. However, FOSS projects can also present hurdles to instructors desiring to take advantage of these opportunities including the learning curve for FOSS culture, infrastructure and processes, difficulties in designing appropriate assignments, and more. This paper presents a model for faculty development based on experiences with prior academic and FOSS approaches to faculty development.
4503	Bridging the academia-industry gap in software engineering: A client-oriented open source software projects course	MacKellar, B.K. ; Sabin, M. ; Tucker, A.B.	Too often, computer science programs offer a software engineering course that emphasizes concepts, principles, and practical techniques, but fails to engage students in real-world software experiences. We have developed an approach to teaching undergraduate software engineering courses that integrates client-oriented project development and open source development practice. We call this approach the client-oriented open source software (CO-FOSS) model. The advantages of this approach are that students are involved directly with a client, nonprofits gain a useful software application, and the project is available as open source for other students or organizations to extend and adapt. This chapter describes our motivation, elaborates on our approach, and presents our results in substantial detail. As we shall show, the process is agile and the development framework is transferrable to other one-semester software engineering courses in a wide range of institutions.
4663	Localized open source software projects: Exploring realism and motivation	Buffardi, K.	Involving computer science students in open source software projects provides opportunities for them to contribute to real products with more authentic scope than typical computer science assignments. However, the environment of collaborating with external, distributed teams also poses unique challenges and may distance students from the potential for valuable, direct contact and mentorship from software professionals. In addition, while the technology industry continues to grow, smaller communities have a vested interest in growing a culture for collaboration between students and local software developers. We formed a local open source organization to collaborate on a product by combining efforts from students and professionals. This paper describes the localized free and open source software (LFOSS) organization and reports initial findings from software engineering students' involvement.
4811	Shaping software engineering curricula using open source communities	Bowring, J. ; Burke, Q.	This paper documents four years of a novel approach to teaching a two-course sequence in software engineering as part of the ABET-accredited computer science curriculum at the College of Charleston. This approach is team-based and centers on learning software engineering in the context of open source software projects. In the first course, teams develop automated testing frameworks for their chosen free open source software (FOSS) project, and in the second course, team members join and contribute to a selected open source project, documenting the process through regular blog and wiki posts and an oral presentation. Results point to the transformative nature of such coursework on students' perception of software engineering. Discussion focuses on potential future iterations of this capstone model.

4815	Collaborative environments in software engineering teaching: A FLOSS approach	Fernandes, S. ; Barbosa, L.S.	Open development has emerged as a method for creating versatile and complex products through free collaboration of individuals. This free collaboration gathers globally distributed teams. Similarly, it is common today to view businesses and other human organisations as ecosystems, where several participating companies and organisations cooperate and compete together. As an example, Free/Libre Open Source Software (FLOSS) development is one area where community driven development provides a plausible platform for both development of products and establishing a software ecosystem where a set of businesses contribute their own innovations. Equally, open and informal learning environments and open innovation platforms are also gaining ground. While such initiatives are not limited to any specific area, they typically offer a technological, legal, social, and economic framework for development, relying always on people as open development would not exist without the active participation of them. This paper explores the participation of master students in FLOSS projects, while merging two different settings of learning: formal and open/informal education.
4966	Team project experiences in humanitarian free and open source software (HFOSS)	Ellis, H.J.C. ; Hislop, G.W. ; Jackson, S. ; Postner, L.	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 FOSS 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 6-year study of students' self-reported attitudes and learning from participation in an HFOSS project. Results indicate that working on an HFOSS project increases 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.

5147	Software engineering learning in HFOSS: A multi-institutional study	Ellis, H.J.C. ; Hislop, G.W. ; Pulimood, S.M. ; Morgan, B. ; Coleman, B.	Real-world projects are frequently used to provide students with professional software development experience. Involvement in Humanitarian Free and Open Source Software (HFOSS) projects allows students to learn about a complex software project within a community of professionals. In addition, the humanitarian aspect of HFOSS provides students with the motivation of developing software that will “do good”. The opportunities for learning in such an environment range from technical topics to communication to professionalism and more. This paper reports on the results of a multi-institution study of student perceptions of learning within an HFOSS project. The study expands an earlier study1 and involves four different institutions with courses offered between fall 2013 and fall 2014. Students were involved in projects including GNOME MouseTrap, a project to provide alternative input for users with disabilities, and OpenMRS, an electronic medical record system used extensively in developing countries. Results generally support the outcomes of the early study, but provide stronger evidence that student involvement in HFOSS promotes student learning in the areas of tools and techniques and technical knowledge about the process and tools used to develop an HFOSS project.
5328	Teaching software verification and validation course: A case study	Mishra, D. ; Hacaloglu, T. ; Mishra, A.	Software verification and validation (V & V) is one of the significant areas of software engineering for developing high quality software. It is also becoming part of the curriculum of a universities’ software and computer engineering departments. This paper reports the experience of teaching undergraduate software engineering students and discusses the main problems encountered during the course, along with suggestions to overcome these problems. This study covers all the different topics generally covered in the software verification and validation course, including static verification and validation. It is found that prior knowledge about software quality concepts and good programming skills can help students to achieve success in this course. Further, team work can be chosen as a strategy, since it facilitates students’ understanding and motivates them to study. It is observed that students were more successful in white box testing than in black box testing.
5329	Lessons learned from teaching open source software development	Morgan, B. ; Jensen, C.	This paper describes student learning within the environment of an HFOSS project that is jointly shared between the GNOME Accessibility Team and three academic institutions. This effort differs from many project-based learning efforts in that the project is shared between the academic participants and the HFOSS community. By involving students in an HFOSS project, learning is started via apprenticeship which allows students to learn from professionals while preparing them for their professional life. Learning within the community of an ongoing FOSS project guides students in the first steps towards understanding the importance of life-long learning as well as providing an initial understanding of the ways in which such learning occurs. The results of a student survey and observations of student reflection papers are discussed.

5335	Learning within a professional environment: Shared ownership of an hfoss project	Ellis, H.J.C. ; Jackson, S. ; Hislop, G.W. ; Diggs, J. ; Burdge, D. ; Postner, L.	This paper describes student learning within the environment of an HFOSS project that is jointly shared between the GNOME Accessibility Team and three academic institutions. This effort differs from many project-based learning efforts in that the project is shared between the academic participants and the HFOSS community. By involving students in an HFOSS project, learning is started via apprenticeship which allows students to learn from professionals while preparing them for their professional life. Learning within the community of an ongoing FOSS project guides students in the first steps towards understanding the importance of life-long learning as well as providing an initial understanding of the ways in which such learning occurs. The results of a student survey and observations of student reflection papers are discussed.
5343	Using a real world project in a software testing course	Krutz, D.E. ; Malachowsky, S.A. ; Reichlmayr, T.	Although testing often accounts for 50% of the budget of a typical software project, the subject of software testing is often overlooked in computing curriculum. Students often view testing as a boring and unnecessary task, and education is usually focused on building software, not ensuring its quality. Previous works have focused on either making the subject of testing more exciting for students or on a more potent lecture-based learning process. At the Department of Software Engineering at the Rochester Institute of Technology, recent efforts have been focused on the project component of our Software Testing course as an area of innovation. Rather than previous methods such as a tightly controlled and repetitive testbed, our students are allowed to choose a real-world, open source project to test throughout the term. With the instructor as both counsel and client, students are expected to deliver a test plan, a final report, and several class-wide presentations. This project has achieved significant student praise; qualitative and quantitative feedback demonstrates both increased satisfaction and fulfilled curricular requirements. Students enjoy the real-world aspect of the project and the ability to work with relevant applications and technologies. This paper outlines the project details and educational goals.
5353	Combining research and education of software testing: A preliminary study	Chen, Z. ; Memon, A. ; Luo, B.	This paper reports a preliminary study on combining research and education of software testing. We introduce some industrial-strength programs, from the open-source projects for research, into the assignments of system testing. Research assistants and teaching assistants work together to establish and evaluate the assignments of system testing. Our preliminary results show that research and education of software testing can benefit each other in this way.
5357	Understanding students'preferences of software engineering projects	Smith, T. ; Gokhale, S. ; McCartney, R.	Students in a maintenance-centric, introductory software engineering course were expected to understand, analyze and extend an open source software project of their choice, selected from a limited set of prepared applications. Students fell into two groups: those who chose a project based on its perceived and estimated difficulty, and those who chose a project based on the appeal of the subject matter. Students in both groups, however, cited value for themselves in terms of enhanced learning experience, and for users in terms of increased benefit, as reasons for their selection. These insights into students' thinking can guide future efforts in selecting projects that can simultaneously support the learning objectives as well as motivate the students, not only in software engineering but also in broader computing courses.

5546	Integrating formal and informal learning through a FLOSS-based innovative approach	Fernandes, S. ; Martinho, M.H. ; Cerone, A. ; Barbosa, L.S.	It is said that due to the peculiar dynamics of FLOSS communities, effective participation in their projects is a privileged way to acquire the relevant skills and expertise in software development. Such is probably the reason for a number of higher education institutions to include in their Software Engineering curricula some form of contact with the FLOSS reality. This paper explores such a perspective through an on-going case study on university students' collaboration in FLOSS projects. The aim of this research is to 1) identify what should be learnt about software development through regular participation in a FLOSS project/community, and 2) assess the didactic potential of this kind of non-standard learning experiences. To this aim we resorted to a participatory research action approach and qualitative methods, namely case studies combining direct observation and interviews.
5676	Enhancing software engineering education through open source projects: Four years of students' perspectives	Papadopoulos, P.M. ; Stamelos, I.G. ; Meiszner, A.	This paper presents the results after four years of running of an instructional method that utilizes free/libre open source software (FLOSS) projects as tools for teaching software engineering in formal education. In the last four academic years, a total of 408 juniors majoring in Informatics (in a 4-year program) participated in the study, assuming the roles of requirements engineers, testers, developers, and designers/analysts. Students appreciated the benefits gained by the method and identified aspects that require further improvement. In the paper, we present (a) the details of our method, (b) students' opinions as recorded through a questionnaire including both closed and open ended questions, and (c) conclusions on how the use of FLOSS projects can be applied, and be beneficial for the students.
17796	Selecting Open Source Software Projects to Teach Software Engineering	Smith, Therese Mary and McCartney, Robert and Gokhale, Swapna S. and Kaczmarczyk, Lisa C.	Aspiring software engineers must be able to comprehend and evolve legacy code, which is challenging because the code may be poorly documented, ill structured, and lacking in human support. These challenges of understanding and evolving existing code can be illustrated in academic settings by leveraging the rich and varied volume of Open Source Software (OSS) code. To teach SE with OSS, however, it is necessary to select uniform projects of appropriate size and complexity. This paper reports on our search for suitable OSS projects to teach an introductory SE course with a focus on maintenance and evolution. The search turned out to be quite labor intensive and cumbersome, contrary to our expectations that it would be quick and simple. The chosen projects successfully demonstrated the maintenance challenges, highlighting the promise of using OSS. The burden of selecting projects, however, may impede widespread integration of OSS into SE and other computing courses.

17800	Teaching Software Engineering from a Maintenance-centric View Using Open-source Software	Gokhale, Swapna and Smith, Th{\'e}r\`{e}se and McCartney, Robert	Software engineering (SE) careers are disproportionately devoted towards maintaining and evolving existing large systems, rather than building them from ground up. To address this focus, we have developed a maintenance-centric SE course that provides students experience in the maintenance and evolution of realistic software projects. For this purpose, we use Open-Source Software (OSS) which is freely available, as a source of realistic software projects. We have found that this course can be effective in exposing students to the kinds of challenges they might face in industry, but that there is a good deal of overhead for the instructor: finding appropriate software projects, designing a set of tasks for the students, and identifying useful (and usable) tools for the students to use. Given a set of projects of appropriate scale and a set of tasks that can fit a semester schedule, we think that this approach could result in an effective SE course that can be offered widely. The purpose of this tutorial is to provide support to instructors to allow them to successfully offer this course without extraordinary preparation.
17805	Teaching Evidence-based Software Engineering: Learning by a Collaborative Mapping Study of Open Source Software	Castelluccia, Daniela and Visaggio, Giuseppe	In this paper, we share our experiences about teaching evidence-based software engineering to students of a Master degree program in Computer Science. We provided a semester-long course, composed of lessons about empirical and experimental methods. It also included a collaborative project concerning a systematic mapping study of the challenges in the adoption of open source software in a business context. All students collaborated on the project by analyzing emerging results in the scientific literature. They evaluated the proposals in terms of level of novelty and evidence and delivered a complete report, which summarized the risk factors in the adoption of open source software and offers technical knowledge about evolutionary patterns and development community support, with practical implications. As a side effect, this problem-based learning approach provides a positive impact in terms of students' participation, teamwork attitude, professional interest in open source software, and exam passing.
17830	A Multi-Institutional Study of Learning via Student Involvement in Humanitarian Free and Open Source Software Projects	Hislop, Gregory W. and Ellis, Heidi J.C. and Pulimood, S. Monisha and Morgan, Becka and Mello-Stark, Suzanne and Coleman, Ben and Macdonell, Cam	This paper reports on a study of student opinion of the impact of participation in Humanitarian Free and Open Source Software (HFOSS) on motivation, computing learning, and major/career direction. The study builds on an existing body of work in student participation in HFOSS. Six institutions with a variety of profiles are involved in the study and the paper reports on quantitative analysis of Likert survey items. Results of Mann-Whitney U tests on Likert data are mixed. Positive results indicate that students perceived that participating in an HFOSS project made them more comfortable with computing and improved their perceived ability to maintain a project and interact with professionals. Negative results include a perceived decrease in perception of computing skills, which may result from an increased understanding of the complexity of developing a large, real-world project.

17845	Student Developed Computer Science Educational Tools As Software Engineering Course Projects	Cicirello, Vincent A.	In a one semester course on software engineering for upper level computer science students, students typically learn the fundamental software processes spanning the software development lifecycle – from requirements specification through architectural design, implementation, testing, and evolution, along with the software tools that support the development activities. Courses in software engineering often incorporate semester long team projects, where students collaborate on a software development project. Thus, in addition to developing the technical skills associated with software development, the software engineering course is a common place where computer science students develop their skills in teamwork and collaboration, as well as in communications. The nature and type of projects in such courses varies. Some integrate projects where students develop software for “real clients” such as on campus departments, or local non-profits, while others have been increasing exposure to open source development with students contributing to existing open source projects. In our software engineering course, we have recently introduced course projects where teams of students specify, design, and implement software that assists computer science students in learning fundamental computer science topics. In this paper, we present our experience with such CS learning tool projects, including a discussion of the impact on student outcomes in a senior-level course on software engineering. We compare this experience to our prior use of “real projects for real clients” in this same course. We find that students who participate in the CS learning tool projects perceive an increase in learning progress on fundamental principles, theories, and factual knowledge, as compared to their peers in course sections with “real projects for real clients,” with an equivalent effect on teamwork and collaboration skills. Most surprisingly, the students who develop CS learning tools report a significantly higher level of progress on industrial relevant skill development as compared to the students who develop so called “real projects for real clients.”
17882	A Collaborative Approach to Teaching Software Architecture	Van Deursen, Arie and Aniche, Maur\'{\i}cio and Au{\'e}, Joop and Slag, Rogier and De Jong, Michael and Nederlof, Alex and Bouwers, Eric	Teaching software architecture is hard. The topic is abstract and is best understood by experiencing it, which requires proper scale to fully grasp its complexity. Furthermore, students need to practice both technical and social skills to become good software architects. To overcome these teaching challenges, we developed the Collaborative Software Architecture Course. In this course, participants work together to study and document a large, open source software system of their own choice. In the process, all communication is transparent in order to foster an open learning environment, and the end-result is published as an online book to benefit the larger open source community. We have taught this course during the past four years to classes of 50-100 students each. Our experience suggests that: (1) open source systems can be successfully used to let students gain experience with key software architecture concepts, (2) students are capable of making code contributions to the open source projects, (3) integrators (architects) from open source systems are willing to interact with students about their contributions, (4) working together on a joint book helps teams to look beyond their own work, and study the architectural descriptions produced by the other teams.

18359	Experience Report: Guiding Faculty Students to Participate in Humanitarian FOSS Communities	Kussmaul, Clifton	Students in computer science (CS) and related disciplines must master content knowledge and skills as well as process skills including communication, critical thinking, problem solving, and teamwork. Free & Open Source Software (FOSS) projects provide opportunities for students to contribute to real software systems and participate in diverse communities, helping students to master both content and process skills. Humanitarian FOSS (HFOSS) projects address social needs, and appeal to many students. However, FOSS can present challenges for students and teachers. To address these challenges, faculty use evidence-based approaches, including Team Project Based Learning (TPBL) and Process Oriented Guided Inquiry Learning (POGIL). This paper describes the redesign of a set of workshop sessions to help faculty learn about HFOSS principles and communities, and how to use HFOSS, TPBL, and POGIL in their own classrooms. © 2016 IEEE.
18433	Helping faculty students to participate in Humanitarian Free Open Source software: The OpenFE OpenPath projects	Kussmaul, Clifton L. and Ellis, Heidi and Hislop, Gregory W. and Postner, Lori and Burdge, Darci	Students in computer science, software engineering, and related disciplines must master a broad range of technical knowledge, skills, tools, and processes. They must also learn to navigate, understand, and contribute to real-world code, documentation, and diverse communities of developers, users, and other stakeholders. One effective way for students to develop such knowledge, skills, and attitudes is to participate in Humanitarian Free & Open Source Software (HFOSS) projects. Research has shown that student participation in HFOSS projects has a positive impact on student motivation to study computing and a strong positive impact on perceived learning related to software engineering [1,2]. The OpenFE and OpenPath projects seek to help faculty and students participate in HFOSS projects and communities. The 2012 NSF TUES OpenFE Project developed and expanded faculty expertise supporting student involvement in HFOSS projects, and created and evaluated learning materials to help students in a variety of settings, including community colleges. OpenFE significantly revised and enhanced the Professors Open Source Software Experience (POSSE), a faculty development program originally developed by Red Hat, Inc. OpenFE also helped faculty to develop, pilot, and disseminate a variety of learning materials [3]. The 2015 NSF IUSE OpenPath Project continues and expands this work with a focus on sequences (pathways) of topics and learning activities that can be integrated across the curriculum to provide faculty and students with a more gradual introduction to FOSS tools and practices. OpenPath also leverages Process Oriented Guided Inquiry Learning (POGIL) [4,5] to help students develop skills in communication, critical thinking, problem solving, and teamwork, which will make them more successful participants in HFOSS [6]. Together, the OpenFE and OpenPath projects have supported over 90 faculty from over 65 institutions to participate in POSSE and explore ways to help their students participate in HFOSS projects. A website (http://foss2serve.org) has more information about OpenFE and OpenPath, POSSE and other events, and learning materials. © American Society for Engineering Education, 2017.