Can Humanitarian Open-Source Software Development Draw New Students to CS?

Heidi J. C. Ellis, Ralph A. Morelli, Trishan R. de Lanerolle, Jonathan Damon, Jonathan Raye
Trinity College
Hartford, CT 06109
011-860-297-4175
[heidi.ellis, ralph.morelli, trishan.delanerolle, jonathan.damon]@trincoll.edu; jr978@bard.edu

ABSTRACT

In this paper, we present an example humanitarian open-source software project that has been used since January 2006 at a small liberal-arts college as an experiment in undergraduate CS education. Sahana (Sinhalese for *relief*) is a free and open-source disaster management system developed in Sri Lanka by a group of IT professionals following the 2004 Asian tsunami. It is a web-based tool that addresses the IT coordination problems that typically occur in trying to recover from a large-scale disaster. We are currently exploring the wider use of Sahana as a sustainable model and platform for teaching about open-source software development while at the same time allowing CS students and educators to make a socially useful contribution of their time, effort, and expertise. This paper presents our experiences with Sahana including the benefits for both academia and industry.

Categories and Subject Descriptors

K.3.2 [**Computing Milieux**]: Computer and Information Science Education - *Computer science education, Curriculum*

General Terms

Human Factors.

Keywords

Open source software, software engineering, curriculum development.

1. INTRODUCTION

In this paper we describe a project that we believe has the potential to address two key questions in computer science education: (1) Can an open-source development project deliver the appropriate kind of "real-world" experience that CS students need? (2) Can a sustainable humanitarian software development effort help CS educators attract a wider range of CS students, especially females, to the study of computer science?

According to ACM President David Patterson and others, there is a disconnect between how programming is taught in the classroom and how cutting-edge software is written in industry. As one way to remedy this problem, Patterson urges colleges and universities to become active participants in the open source movement [12].

We agree with Patterson's assessment and have been searching for ways to incorporate open-source education into the CS curriculum. With enrollments in CS near their all-time lows, can adoption of the open-source movement help make the study of computer science more relevant to the current reality?

Part of the problem with low enrollments is the widespread misconception that outsourcing is decreasing the number of IT positions in the U.S. But the U.S. Department of Labor, Bureau of Statistics forecasts that software engineering positions will be among the fastest growing jobs over the next 10 years [7]. In order to meet this growing demand, CS educators must make the study of computing and software engineering more attractive and more relevant to today's students. Can the open-source movement help us in that endeavor?

Another well known problem in academic computer science is the under-representation of women students [2]. One explanation given for this problem is that women are less interested than men in learning programming for its own sake and more interested in outcomes that can be achieved through programming. For example, researchers who have studied women's attitudes toward computing have indicated that women are attracted to computing for its social value and are interested in the intersection of computing to other disciplines such as education, medicine, and the arts [5].

In a recent column addressing the Hurricane Katrina disaster, Patterson raised the questions, "Should computer scientists and engineers take on greater responsibility to help reduce the loss of life and property damage from natural disasters?" [11] We find this question intriguing not only as a powerful way for computer scientists to contribute their expertise in a socially useful way, but also as a potential means of giving meaning and purpose to the process of learning to program. If programming for its own sake is a "turn off" to some students who have shown interest in CS, might programming for humanity's sake help us "turn on" these students to the joys of helping to create good and useful software?

The Trinity Sahana project is an open-source humanitarian software development effort that aims to design and build software that can be used to assist in large-scale disaster recovery. It is part of an international project, begun in Sri Lanka following the 2004 Asian tsunami. As CS educators and students, when we first heard about Sahana, we were interested in whether

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCSE'07, March 7–10, 2007, Covington, Kentucky, USA.
Copyright 2007 ACM 1-59593-361-1/07/0003...$5.00.

participation in the Sahana project could help us incorporate a much-needed "real world" software engineering component into our curriculum and, at the same time, help us attract more students and a greater diversity of students to the study of computer science.

With the help of a summer research student, and with volunteer help from faculty, staff, corporate associates, and a visiting student, we conducted a pilot study over the summer during which we developed a dual-purpose module that has now been incorporated into the Sahana platform. The pilot succeeded in demonstrating the feasibility of developing significant open-source software with undergraduate programmers and taught us many positive (and some negative) lessons about this kind of educational experience. We see good potential for this type of project both to introduce our students to real-world software development practices and to generate excitement about programming. Therefore, we are in the process of expanding its scope to involve students from nearby colleges and universities.

2. THE OPEN SOURCE MOVEMENT

The open-source movement has grown exponentially. SourceForge, one of the primary hosting sites for open-source software, lists 127,947 registered projects and 1,377,712 registered users as of August 2006 [14]. Although the open-source movement was started and incubated in university and academic settings, interest in it has been growing rapidly among commercial and corporate software developers and users. For example, the 2006 Open Source Software Convention (OSCON) lists some of the leading software and internet companies as its sponsors, including IBM, Microsoft, Oracle, Sun Microsystems, Google, HP, Dell, Intel, and Yahoo! [8]. This widespread and growing interest in the open-source model is strong evidence that the model is fast coming to be recognized as both a commercial and technological success.

2.1 Open-Source and CS Education

Given the tremendous interest in open-source within the software industry, one would expect there to be a similar burgeoning of interest in the CS and IT educational communities. However there appear to be few reported projects that have successfully demonstrated how to incorporate open-source development into the undergraduate curriculum. A panel presentation at the 2002 SIGCSE characterized open-source as a challenge to the *status quo* and debated some of the intellectual and ethical challenges involved [15]. Surprisingly, one of the panelists argued that because open-source software (OSS) lacked appropriate standards, it should not be used as a model for software development.

A 2002 project investigated the use of OSS in the learning, teaching and development of educational robotics, in particular the use of LEGO Mindstorm's *Robotics Invention System* [10]. Carrington and Kim describe a very successful looking course that used the open-source model in teaching a software design course [3]. The final project for the course required teams of students to extend an open-source software tool that they had spent the semester studying. Shockey and Cabrera have created an ongoing open-source educational and research project at the Inter-American University of Puerto Rico [13]. This ambitious project has led to the creation of an open-source Java-based software development platform and center, funded in part by the Puerto Rican government and run with the help of undergraduates. In the Dr. Java project, Allen, Cartwright, and Reis demonstrated the feasibility of using open-source software tools and the extreme programming methodology to develop production quality software in the classroom [1].

In addition to these classroom-based projects, the Open Source Development Labs website maintains links to ongoing open-source projects in higher education [9]. OSDL currently lists 16 projects based at several colleges and universities in the US, Europe, and Asia. Among the various projects, Portland State University lists a course titled *Open Source Software Development in the Linux Environment* [6]. Another exemplary project is the Alice project at Carnegie Mellon University. Alice is an introductory programming language and environment that is currently being developed through a partnership between CMU and Electronic Arts, Inc. [4].

Despite the relative paucity of projects that have attempted to use the open-source models in the software engineering classroom, a number of important lessons can be drawn from these earlier efforts:

• Because of inadequate documentation, poor coding style, and other "barriers to entry", care must be taken to select a suitable open-source project [1].

• Proper methodologies—such as extensive documentation, precise requirements, re-factoring, and unit-testing—must be used to make the experience accessible to student programmers [1].

• The use of easy-to-use open-source development tools is an important factor in student success [3].

• The choice of appropriately scaled goals is an important determinant of success [13].

2.2 Humanitarian Open-Source Software

The project we describe here has certain similarities with the classroom-based projects described in the preceding section. Thus, our goal is to teach software engineering principles and practices while contributing to an ongoing open-source development project.

Our project differs from the Dr. Java project [1] in that we do not completely control the development of the Sahana software. The Sahana project, as incorporated into Trinity's curriculum, has an external, real-world client in the Sahana core team and the participating non-profit organizations. As such, development, deadlines, and deliverables must conform to the Sahana standards. Requirements are developed by students and reviewed by Sahana team members. All code produced by students undergoes a comprehensive code review by the Sahana core team before it is incorporated into Sahana. Sahana core team members and participating industry and non-profit partners perform alpha testing. In addition, students must learn to operate in a distributed development environment and to use standard communication and documentation tools such as a wiki and discussion lists. We do note that the open-source nature of the Sahana project provides a degree of flexibility not typically found in a commercial development environment, but the rigor of the development process is equivalent. Therefore, open source offers the precision of a commercial environment yet the flexibility needed in an academic environment.

Another distinguishing characteristic of the Sahana effort is that it can be used as an open-source software platform both within the confines of a semester or course and as an ongoing project. Because the needs of the project vary so widely—from software design, to coding, to documentation, to testing—it is open to CS and IT students at various levels of their careers.

Finally, in addition to being a "real world" programming project, the Sahana project's humanitarian dimension makes it potentially attractive to students who see programming as a means to solving socially important problems rather than as an end in itself. Our hope is that this aspect of the project will make it attractive to female students.

3. THE SAHANA PROJECT

Sahana (Sinhalese for *relief*) is an open-source humanitarian software system that was started in Sri Lanka by a group of IT developers (http://www.sahana.lk/). It is a web-based tool that addresses the IT coordination problems that occur during recovery from a large-scale disaster. Sahana has been deployed in a number of natural disasters, including in Sri Lanka during the 2005 Tsunami, during the 2005 earthquake in Pakistan, during the 2006 mudslide in the Philippines and most recently during the 2006 Jakarta earthquake. The Sahana project has received numerous awards for its humanitarian efforts.

3.1 Trinity Team Sahana Module

The Trinity Sahana project began in January 2006 following a serendipitous meeting between the CS department's technology coordinator and core Sahana team members at the University of Colombo School of Computing, Sri Lanka.

Based on an initial assessment of the viability of the project and expected student interest, an independent study course was run during the spring 2006 semester. During the semester, two Trinity undergrads worked with a professor and an IT professional to determine the feasibility of working within the Sahana framework and its usability for undergraduate computer science students without prior experience working with the Linux/Apache/MySQL/PhP (LAMP) architecture. Students gained an understanding of Sahana's architecture and gained facility in working in an integrated programming environment.

Feedback from this independent study indicated several things. It was apparent that the Sahana framework provides a solid scaffold upon which undergraduate education can be structured. In addition, it was apparent that Sahana is a project with the potential to provide students with a unique opportunity to work on a real world software system, with tangible results. However, it was also clear that a set of requirements was needed in order to guide development.

During the latter part of spring 2006, requirements for a volunteer registry/management module to be added to Sahana were developed in conjunction with the Accenture Corporation, and with Points of Light and Aid Matrix, non-profit humanitarian organizations. The intent of the module is to support the registration of volunteers and their assignment to projects.

At the end of the spring 2006 semester, funding was secured to sponsor a summer research student at Trinity to work on the Sahana project. At this point the group was expanded to include an additional faculty member and a student intern from Bard College. A prototype of the volunteer registry/management module was developed, taking just under three weeks to develop and comprising 6000 lines of original source code.

The volunteer registry/management module was presented at a workshop at the National Conference of Volunteering and Service in Seattle on June 20th, 2006 as part a demonstration of the Sahana system. The module was demonstrated to representatives from various national and international volunteer organizations, many of whom expressed surprise that Sahana was open source and therefore free. The remainder of the summer was spent re-factoring the code in the module and incorporating additional features identified from feedback received during the conference.

In August 2006, the Trinity Sahana group was asked to support the Sahana core team's involvement in the Strong Angel III Disaster Response Demonstration in San Diego (strongangel3.net). Members of the Trinity team traveled to San Diego to work in collaboration with the lead contributor to the Sahana project and other members of the Sahana team remotely to demonstrate the volunteer registry/management module, along with the other modules in Sahana. In fact, our team's volunteer registry/management module was used to register all volunteers in the Strong Angel III event.

4. LESSONS LEARNED

4.1 Student Thoughts

The students who have worked on the Trinity Sahana project over that past several months have been very positive about the open-source experience. In the words of one student,

> "Like most CS students, we had never been contributors to OSS, and working on Sahana was something of a revelation. Typical programming assignments begin with a blank screen. But in this context, there was a huge repository of pre-existing code, so writing a new page almost inevitably involved modification of an old one. At first surprising and later gratifying, these modifications could then be observed right where they would end upon a World Wide Web page."

Some of the lessons that emerged from the project in the students' own words were:

Documentation. It is important to write good documentation for all of the code that is produced. Not only did good documentation serve to refresh one's own memory, it was a necessary reference for all the software experts who would be using the code, the sort of people we don't want to disappoint. Thankfully, the process of writing and reading documentation for Sahana was greatly simplified by the use of the PHPDoc tool.

Self-documenting Code. Writing smart, self-documenting code is an unstated project requirement. Because we were coding for a broader and more demanding audience, the clever or shoddy solutions that ordinarily pass muster in a CS class are simply not welcome in the OS environment. In order for our contributions to be accepted, it was necessary that they possess the right functionality and achieve it the right way.

Further, thrashing through dense or poorly documented pre-existing code was a learning experience in itself. We were forced to see how the code worked, often times experimentally. At the same time, we would be learning how **not** to write our own programs.

Real-world Development Environment. Working with the Sahana project, we needed to find our own place in a real world organizational network. This may mean changing deadlines, changing specifications, database updates -anything, really. Unlike an independent project, which is purely one-dimensional (receive the programming assignment and do it), an OSS project is multi-dimensional. It is necessary to know where our contribution is needed, and what is needed of our contribution. This may involve keeping in close contact with the core team: receiving updates, doing teleconferences, sending questions. It may also mean collaborative, student-to-student problem solving. The glitch that has you stumped may be easy going for your peer across the aisle.

Professional Contacts. Members of the Trinity team that traveled to San Diego for the Strong Angel III event made many professional contacts. One student worked one-on-one with developers from Microsoft Corporation and Google. The student also interacted with people from Accenture and IBM. It is expected that the relationships established during this event will continue long afterwards.

As these observations suggest, students involved in the project were able to appreciate first hand the importance of software engineering concepts that are often difficult to convey in a formal course. Because it was going to be published with their names attached, they took enormous pride in their code and documentation. Because they were in a fluid and sometimes chaotic development environment, they had to be flexible and find their own ways of fitting in. Because they could see the utility of the product they were creating, they took special care in getting the design right and solving unanticipated problems that arose. As CS educators, these are the kinds of responses we always hope to see but only rarely achieve in the classroom.

4.2 Faculty Observations

From the faculty perspective, the pilot project was an enormous success and we are now planning to expand its scope and incorporate it into our curriculum during the coming year. Development of open-source modules proved to be an excellent way to provide students with real-world experience. Since the pilot project was part of a true open-source effort, we were forced to operate within the very fluid and sometimes chaotic constraints that come with that environment. The development team was diverse and distributed and included representatives from industry and from social service organizations. The project involved interacting with real clients that we had never met face-to-face and dealing with a schedule and a set of requirements that changed on the fly. It required the students to deal with not very well documented code written by other programmers. However, despite these realities, the students in the pilot project were able to design and produce code and test deliverables that had to meet professional quality standards.

A number of lessons emerged from the project.

Communication. Given the various different people involved in the project, good communication is necessary to keep all members up to date. Frequent team and group meetings, including occasional conference calls with clients and sponsors, are important. Similarly, use of electronic forms of distributed communication—email lists, technical discussion groups—are an essential element of designing, testing, and gaining approval and adoption of open-source software. These are forms of communication that have become standard in the software industry but are less frequently used in the computer science classroom.

Open–Source Tools. It is important to select carefully the open-source software used to support the project. In addition to being functional the tools have to be accessible to students. Our toolset included the LAMP set of Linux, Apache, MySQL, and PHP. One type of tool that we did not use on the pilot project was code-versioning software. With such a small group of coders and developers we were able to manage different versions of the system by hand. But as the project grows, it is clear that that a source code control tool will be necessary.

Faculty Development. One of the most enjoyable aspects of the project was the experience of faculty and students learning something new together, more or less as equals. Faculty members don't typically get the opportunity to participate in a real-life project and learn new tools and solve new problems together with students. We found such collaborative learning to be a healthy and enjoyable pedagogical experience for students and faculty alike.

5. A SUSTAINABLE MODEL FOR CS EDUCATION?

In terms of the general implications of an open-source humanitarian project for CS education, we believe that this or a similar project could help address some of the problems that presently plague our discipline.

Join the Open Source Movement. Establishing or joining a wide-ranging project such as Sahana is one way to introduce students and faculty alike to the open-source movement. In addition to the bonding and teamwork it encourages between faculty and students, the extra-curricular nature of the project allows a department to experiment with open-source in ways that can be difficult to carry out in the classroom. It also opens up relationships and contacts with industry that will be helpful to students in their search for employment.

Help Our Neighbors. Community service is an important part of undergraduate education at many colleges and universities. However, computer science students have sometimes lacked for ways to make meaningful contributions within the community. Helping to build free humanitarian software for use in disaster recovery provides a unique and appropriate way for CS students to contribute. Sahana or a similar project would provide a new channel of volunteerism for computing students, enabling them to contribute their time and expertise toward a humanitarian project that has tremendous potential impact. Moreover, the contributions can be made on a local as well as international scale. For example, we are currently talking with local social service organizations about using Sahana's volunteer registry/management module in their organizations.

Broaden Interest in the Discipline. A project like Sahana, with its emphasis on building socially useful software, may hold significant potential for attracting women students and others who have avoided studying computer science because they fail to see that one of its chief tools—computer programming—can be used to solve socially significant problems. Over the next year we hope to be able to test this hunch by making a concerted effort to publicize the project among our introductory students, both major and non-majors alike.

Strengthen Town and Gown. The Sahana project can help strengthen the relationship between academic departments and IT companies. In the short time we have been engaged in the project companies such as Accenture, IBM, Microsoft and others have shown varying degrees of interest and involvement. Companies appear willing to provide various forms of financial, logistical, and mentoring support for Sahana activities. Computer science students get to meet and, in some cases, work with IT professionals, making invaluable contacts in the process and gaining an opportunity to see what a career in the software industry or in a particular company would look like. By supporting such projects, IT companies can create positive intern-like relationships with promising future employees.

6. FUTURE PLANS

While the initial outcomes of using the Sahana project in an undergraduate computer science program have been quite positive, we still lack longitudinal experience to answer the questions of whether a humanitarian open-source development project can deliver "real-world" experience and will it attract a wider range of CS students. Clearly the ideas generated about the potential benefits of open-source, humanitarian software development are unproven and need further study. We are currently in the process of planning such a study. Our plan is to incorporate open-source software development into the academic curriculum through the offering of two courses. In the fall 2006 semester, a joint independent-study, for-credit open source Sahana project involving teams from participating schools (initially Trinity College, Bard College, Wesleyan University, and Connecticut College) will be offered. A follow-on course will be offered during the spring 2007 semester. The course will be a distributed video-conference course on "Open source software development" involving participating schools. In both of these course offerings, students, faculty, and other contributors from these schools will participate jointly in development of one or more open-source disaster management modules.

In order to provide sustained support of the Sahana project across multiple academic institutions, we plan to establish a collaborative space for various groups to develop software solutions. This space may include a source code repository and versioning system, a wiki, and collaborative software.

Among the questions we will address in the ongoing study are:

• Does participation in real-world open-source software development improve the educational experience and career prospects of computer science majors?

• Does the humanitarian component of the project help attract students who would otherwise not be interested in computer science?

• What specific guidelines should be adopted concerning software tools, development approaches, project guidelines, and other practical issues involved in such a project?

7. REFERENCES

[1] Allen, E., Cartwright, R., and Reis, C. Production programming in the classroom. *SIGCSE 2003,* (Feb. 2003), 89-93.

[2] Bombardieri, M. In computer science a growing gender gap: Women shunning a field once seen as welcoming. *Boston Globe*, December 18, 2005, http://www.boston.com/news/local/massachusetts/articles/2005/12/18/in_computer_science_a_growing_gender_gap/?page=full, retrieved August 11, 2006.

[3] Carrington, D., and Kim, S. K. Teaching software design with open source software. *ASEE/IEEE Frontiers in Education Conference* (Nov. 2003), 9-14.

[4] Carnegie Mellon collaborates with EA to revolutionize computer science education, http://www.cmu.edu/cmnews/extra/060310_alice.html, retrieved August 27, 2006.

[5] Cuny, J., and Aspray, W. Recruitment and retention of women graduate students in computer science and engineering: Report of a workshop, June 20-21, 2000. http://www.cra.org/reports/r&rwomen.pdf, retrieved August 11, 2006.

[6] Massey, B. PSU CS410: Open source software development. http://ossclass.wiki.cs.pdx.edu/FrontPage (Summer 2006), retrieved August 27, 2006.

[7] Occupational Outlook Handbook, http://www.bls.gov/oco/oco2003.htm, retrieved August 11, 2006.

[8] OSCON Open Source Software Convention, http://conferences.oreillynet.com/os2006/, (July 2006), retrieved August 27, 2006.

[9] OSDL Higher Education OSS Projects, http://groups.osdl.org/forums/higher_ed_projects/, retrieved August 27, 2006.

[10] O'Hara, K. and Kay, J. Investigating open source software and educational robotics. *Consortium for Computing in Colleges* (2002), 8-16.

[11] Patterson, D. Rescuing our families, our neighbors, and ourselves. *CACM, 48, 11* (Nov. 2005), 29-31.

[12] Patterson, D. President's Letter. *CACM, 49, 3* (Mar. 2006), 27-30.

[13] Shockey, K. and Cabrera, P. Using open source to enhance learning. *ITHET 2005* (July 2005), 7-12.

[14] SourceForge.Net, http://sourceforge.net/, retrieved August 27, 2006.

[15] Wolf, M, Bowyer, K, Gotterbarn, D, and Miller, K. Open source software: Intellectual Challenges to the Status Quo. *SIGCSE 2002* (Mar. 2002), 317-318.