Holistic Software Engineering Education Based on a Humanitarian Open Source Project

Heidi J. C. Ellis, Ralph A. Morelli,
Trishan R. de Lanerolle
Trinity College
{heidi.ellis, ralph.morelli,
trishan.delanerolle}@trincoll.edu

Gregory W. Hislop
Drexel University
hislop@drexel.edu

Abstract

For the past year, Trinity College has utilized Sahana, a free and open source disaster management system, as a foundation to teach software engineering. The goals of the use of the Sahana project are threefold: to provide students with a real-world software engineering experience; to introduce students to the open-source development model; and to attract a wider variety of students into computing due to the real-world and humanitarian nature of the Sahana project. This paper discusses an approach for using open source software as a foundation to teach software engineering in a Liberal Arts environment by involving students in an ongoing, real-world project from the very beginning, allowing students with a wide range of backgrounds to participate. Results of a learning survey of a small group of students who have participated in the project are presented. The paper also provides guidance to others contemplating incorporating open source projects into their software engineering courses or curriculum.

1. Introduction

The need for software engineering students to gain real-world experience during their academic studies is well known [5], [15]. The trend of software production towards globally distributed development that increasingly uses open-source components and components off the shelf (COTS) highlights the need for students to be prepared to work in a distributed environment [6]. The open-source movement has been gaining momentum within the software industry [13] and has been suggested as one way to provide students with industry experience in an academic setting [12].

For the past year, Trinity College has been using Sahana [2], an open-source disaster management software project, as the foundation for software engineering education. Trinity is a competitive Liberal Arts college. Sahana, Sinhalese for relief, is a free and open-source disaster management system (http://www.sahana.lk). It is a web-based tool that addresses the information technology (IT) coordination problems that typically occur in trying to recover from a large-scale disaster. Sahana is part of an international project that began in Sri Lanka following the 2004 Asian tsunami and has been deployed in a number of disasters, including the 2005 earth quake in Pakistan and the 2006 mudslide in the Philippines. At Trinity, undergraduate students are developing production-level software which has been incorporated into the most recent Sahana release. The educational goals of the effort are: to provide students with a real-world software engineering experience; to introduce students to the open-source development model; and to attract a wider variety of students into computing.

1.1 Open Source Software in Software Engineering Education

Interest in open-source software is growing in the software education community. Many institutions have adopted the use of open-source software such as Linux, PhP, Apache, and

others as application tools. However reports on the use of open-source software as a basis for software engineering projects are just starting to emerge.

In one of the earliest efforts, Carrington and Kim describe a software design course that uses an open-source project [3]. Student teams were required to evaluate and extend an open-source software tool during the progress of the course. Similarly, Toth [16] discusses the use of open source software tools as the basis for a capstone course. Toth describes a two-term practicum where students evaluate existing software engineering tools, select a candidate tool to enhance, and extend the functionality of the selected tool.

Allen, Cartwright and Reis [1] outline the use of open-source software as the basis for a software engineering course intended to provide students with production-level development experience. The course revolves around the enhancement of the Dr. Java tool, an integrated software development environment for Java. Shockey and Cabrera [14] discuss an ongoing education and research project at the Inter-American University of Puerto Rico which has led to the creation of an open-source Java-based software development platform and center where undergraduate students can gain experience in open source development.

Liu [10] presents an open-source software development process called Gradually Ripen Open-source softWare (GROW) that is planned to be used in a project-based software course. The goal is to have teams of students working simultaneously in various stages of development on real-world projects. Unfortunately, no results of the application of the GROW process have been reported.

The work described in this paper differs from those described above in two important ways. First, many of the efforts focus on open source projects developed for an audience of software developers whereas Sahana is humanitarian software, intended for use by non-profit and non-governmental organizations. Second, the Sahana effort at Trinity College is taking place in an open-source community that has an integrated mix of people, some of whom are working on the project as volunteers and others for credit.

2. Sahana in software engineering education

There are several characteristics of the Sahana project that make it an ideal foundation upon which to base software engineering education. The Sahana project is developed using the open source LAMP architecture (Linux, Apache, MySQL, and PhP) and currently contains over 70,000 lines of code. Therefore Sahana provides an existing application of considerable size and complexity upon which education can be based.

Sahana is an ongoing project where software functionality is continually being expanded and enhanced. By involving students in contributing to a production version of the Sahana project, students are exposed to a real-world development process including requirements definition, reviews, adherence to coding standards, formal testing, etc.

The Sahana core development team, located in Sri Lanka, is supported by a global volunteer community of humanitarian consultants, emergency management experts and developers, numbering in excess of 150. The distributed nature of the Sahana project provides students with a real-world experience of dealing with a variety of developers and customers spread around the world, personnel not typically found in the traditional academic environment. Students must also become familiar with the distributed development tools.

The use of Sahana, or any disaster relief software, is characterized by high modifiability demands on the software. Given the very unpredictable nature of disasters, the Sahana project must be quickly and easily modified along defined axes (e.g., customization of volunteer registration interface). The strong requirement for adaptive maintenance provides students with the real-world experience of changing requirements.

2.1 Educational Approach

The general approach to using Sahana as a base for software engineering education is to incorporate students with a range of backgrounds and experience into a real-world project where each student contributes based on their knowledge and abilities. For instance, students with less software engineering background may begin by eliciting requirements or updating project documents while students with more extensive software background may start by designing portions of an application. Small classes and independent studies are used to provide individual attention and support the software engineering team experience.

Trinity began investigating the use of Sahana for software engineering education in January 2006 through several independent study courses and summer internships as described in [4]. The focus of the early work was on the development of a volunteer management (VM) module for Sahana. During fall 2006, an independent study course was offered at Trinity in a distributed format with participation by students from two other Liberal Arts colleges, Wesleyan University and Connecticut College. Several other college-age individuals also volunteered for a semester of effort without receiving course credit. The minimal background for entry into the course was CS1 and students with a wide range of backgrounds took the course. The course attracted a total of nine students with two each at Wesleyan University and Connecticut College. Six of the students took the course for credit and three were volunteers.

The participants somewhat self-organized into teams and roles with some students working on developing enhancements to Sahana and others serving as infrastructure support such as project manager, resource coordinator and librarian. Students were allowed to identify their own projects and students initiated three different efforts into enhancing Sahana.

The two students at Wesleyan University developed a hospital management module to keep track of available medical facilities in the area of a disaster. The module enables users to locate hospitals within a particular region and to manage the status of hospitals (e.g., number of total beds, number of available beds, etc.). This project is ongoing as the students intend to continue their efforts during the spring 2007 semester.

The two students at Connecticut College developed an extension to Sahana to allow the bulk import of data into the Sahana missing-persons database. The application is intended to allow an entire college or corporation to quickly and easily set up a missing persons registry of everyone in their organization. Supported upload formats include LDAP, XML, and CSV. This project was requested by the Connecticut College disaster planning group and is being incorporated into the campus plan for disaster management.

Two Trinity students worked with Trinity's Community Service and Civic Engagement office to adapt the VM module to provide a customized interface that could be used to register volunteers for a variety of community volunteer projects. The idea behind this project was to make the VM interface more general and customizable to serve community organizations working outside the domain of disaster management.

2.2 Process and Procedures

The course met one evening a week for 15 weeks using videoconferencing. Potential enhancements to Sahana had not been identified before the start of the course and therefore the first few weeks of the course were spent identifying projects. This somewhat slow start to the course meant that more time was spent on the initial software engineering phases of requirements and design.

The Sahana effort dictates coding standards, but does not provide any other documentation standards. In addition, there is little in the way of prescribed development process other than that used by the core team to approve additions. Therefore a series of project documents were provided to students as research has shown that such scaffolding is beneficial for providing structure for both the course and for the software development process used in the course [7].

All of the projects ran on the same schedule. After identifying a project, students completed a software requirements specification (SRS) loosely based on the IEEE-830 [8] and slightly modified from the work presented in [7]. Students were provided with an example of the requirements specification developed for the VM module. During the sixth week, students presented drafts of their requirements during class and any feedback was incorporated into the final document which was submitted in the seventh week of class.

After an initial review and revision of the SRS, students went on to complete a software design specification (SDS), loosely based on the IEEE-1016 [9] and modified from [7]. The SDS template was explained in class and an example SDS from a non-Sahana application was provided. During the 10th week in the semester, students presented drafts of their SDS documents. Comments resulting from this informal review were incorporated and final versions were submitted in week 11.

During the remaining four weeks in the semester, students focused on implementing their designs and no further documentation was required. Students presented their projects to faculty from all three schools as well as to members of the Sahana community and industry representatives who attended the meeting via video conference.

Grading was done collaboratively by all participants in the effort. The SRS, SDS, and final presentation were the only graded deliverables and all participants used a common rubric to grade each team's efforts. The high and low grades were eliminated and the grade for the deliverable was calculated by averaging the remaining scores.

In addition to the documentation scaffolding, the course was also supported by a wiki (http://www.cs.trincoll.edu/sahana_proj/wiki/Main_Page). Students were encouraged to collaborate both within their team and across teams. The result was a true community effort in creating a useful wiki which contains a set of examples, FAQs and a series of developer guides. The Sahana core team found the documentation developed on the wiki to be so useful that they included the new developer guide in their main documentation.

3. Software Engineering Learning

3.1 The Survey

An anonymous survey instrument was constructed to elicit student observations on their learning during the fall 2006 semester. The survey contained four sections. The first section asked for student background while the second section contained free-form questions pertaining to student learning. The third portion of the survey asked students to rate their learning about the software engineering topics from the SWEBOK based on a five-point Likkert scale. The last portion of the survey asked students to indicate their satisfaction with the course using a five-point Likkert scale using a variety of statements about the course.

3.2 Results and Observations

There were nine participants in the fall 2006 Sahana effort, few of whom had any formal software engineering background. Of the five students who returned survey responses, all were male, all had taken the course for credit, and all were either juniors or seniors. The technical background ranged from two semesters of Java to strong PhP and MySQL experience. Motivation for taking the course ranged from interest in Sahana to interest in developing a web application. One student indicated an interest in working on a real-world project: "I was excited to work on something that had the potential for real-world use, rather than just class exercises."

Students indicated that the primary knowledge gained from the course ranged from tools to software design to software development process and working in a team. A student said: "As I was working with others, I had to make considerations about other's work and make it

compatible with mine." The top roadblocks to student learning identified by students included the steep learning curve for some aspect of the project, and technology blocks including incomplete library documentation, lack of JavaScript skills, and difficult-to-trace bugs. In answer to the question about how much students felt they learned about software engineering, "a fair amount" was the common answer.

Table 1. Self-Assessment of Learning Based on SWEBOK Categories.

Topic	None	A Little Bit	Some	A Lot	A Significant Amount
Software Requirements			1	3	1
Software Design			2	1	2
Software Construction			1	1	3
Software Testing		3	2		
Software Maintenance		2	2	1	
Software Configuration Management		1	3		
Software Engineering Management	1	1	2		1
Software Engineering Process		2	1		2
Software Engineering Tools and Methods	1		2	1	1
Software Quality		2	2	1	

Table 1 shows the student self-assessment of learning based on the SWEBOK categories. The results clearly show that students felt that they accomplished some software engineering learning. In addition, most students felt that they learned an appreciable amount about the early phases of software engineering including requirements, design, and construction. As could be expected, these results reflect the greater emphasis placed on these topics during the course. Similarly, students indicated less learning occurring in the latter phases of development such as testing and maintenance. It should also be noted that while a brief description of each of the terms was included in the survey, some students may not have clearly understood the terms.

Table 2. Student satisfaction with the independent study.

Question	Mean
21. I have a high level of interest in the course subject matter.	3.75
22. The subject matter of this course is highly relevant to my future career plans.	3.75
23. I have a high level of experience in the course subject matter.	2.50
24. I like the mix of theoretical learning combined with hands-on application of the subject matter.	3.50
25. The pace of the course was too fast.	2.50
26. The pace of the course was too slow.	2.25
27. I am very satisfied with my learning experience in this course.	4.00
28. Overall, I am very satisfied with this course.	4.25

The results from part four of the survey on student satisfaction were mapped to integers with one representing the "strongly disagree" answer and five corresponding to the "strongly agree" answer. Table 2 shows the mean student responses. The slightly positive responses to questions 21 and 22 indicate relatively high student interest in the course. The below-average mean of the responses to questions 25 and 26 indicate that students appeared to be satisfied with the pace of the course. The strong positive responses to questions 27 and 28 indicate that students were very satisfied with their learning. During the final presentation of their projects, students were asked about the "best" thing that they had learned. The Connecticut College students said that the most important things they learned were how to work in teams and the importance of

documentation. The Wesleyan team reported that they learned about how to understand the project by reading code and project documents.

4. Discussion

Observation of students working on the Sahana project have yielded several findings about student learning. One result that we have observed is that students quickly learn about requirements. The first task students have is to understand the Sahana system and then to determine boundaries of their particular project. During this process, students learn about the negative impacts of incomplete or inadequate requirements documents. One complication in eliciting requirements, also identified by McGrath [11], is the difficulty in deciding on requirements with a distributed group of users with varying and sometimes conflicting needs.

Once students have identified a project, they must then understand how their project fits into the larger Sahana effort. During this process students learn design by examining the existing structure of the Sahana system. While not perfect, the Sahana system provides an example of solid design. In addition, examination of the existing system highlights the coding standards used in Sahana.

Students gain an understanding of software quality through the use of periodic reviews. All project documents are jointly reviewed by the class and feedback is provided. In addition, students are motivated to test rigorously as professional developers will be testing their code. They also have the further incentive of knowing that if their code fails, the failure could have significant impact on human life.

The changeable nature of the Sahana project underscores the need for estimation, risk assessment, and scheduling. The potential for change is very high and this requires students to understand the impact of change and how to plan for change.

The instructors have observed that as students have gained more software engineering knowledge and experience, their contributions have increased in volume and quality. In addition, as student knowledge increases, they take on greater responsibility for the project. In this manner, students grow in software engineering experience and professionalism.

4.1 Benefits and Drawbacks

There are many benefits reaped by both students and instructors from working with an open source humanitarian software. Benefits that result from the open source nature of the project include increased industry contacts, professional maturation of students, improved written communication skills, experience with system administration, and exposure to management approaches [15]. Similarly, Toth notes benefits which include having access to an existing project of significant size, high student motivation, increased interaction with industry, and increased marketability of students [16].

We have observed all of these benefits in students who have worked on the Sahana project. The benefits that students mention most are those related to real-world experience and marketability. We have seen students more quickly mature as professionals due to increased exposure to real-world conditions and increased interaction with software professionals.

Another benefit obtained from an open source project that we have observed is that software engineering principles are conveyed by experience rather than by the instructor’s voice. While the instructor introduces software engineering precepts, the principles themselves are embedded within the project and make themselves evident to the students as they go through the planning and development process. As a result, the instructor is more of a guide and students gain a broad understanding of the precepts.

In addition to the open source nature of the Sahana project, the humanitarian aspect of Sahana also provides some benefits to students. The involvement of professionals, both as customers and as fellow developers, provides valuable role models for students. Students also benefit from increased professional contacts from a wider range of organizations as well as improved team interaction. One of our students has worked side-by-side with professionals from Google, Accenture, IBM, and the Red Cross.

While there are many advantages to using humanitarian open source software as a base for software engineering education, there are also several drawbacks. Our experience is based on a few small teams supported by two instructors and another Sahana volunteer. Based on our experience, we expect that managing a project with a large number of students could be unwieldy if students bring a wide variety of backgrounds to the course.

A second drawback to our approach is difficulty in grading due to the different background and experiences that students bring to the course. Since students make different contributions to the project, there is no uniform metric that can be applied to all students. A grading drawback in the peer-grading approach was that students tended not to be as constructively critical of each other as we might have hoped.

Another significant drawback resulting from students contributing to the production version of Sahana is the case where the project is deployed for use in a disaster in the middle of a semester. Such an event would require that some current development cease and effort be put into supporting the disaster (i.e., customizing and maintaining the software for the disaster). Such a situation requires flexibility on the part of the instructor to change the direction of the course in mid-stream.

4.2 Guidance for Using Open Source Projects in Software Engineering Education

The use of an open source project as a base for software engineering education has many advantages, but can also have disadvantages including lack of documentation, inconsistent coding standards, and inconsistent quality. Therefore, we suggest that such efforts start small in order to allow instructors time to understand the project as well as to understand the approaches that best merge open source development and academic requirements. Indeed, Toth [16] notes the use of open source software in an academic course requires an investment of instructor time. Therefore a small number of small teams would provide an ideal starting point.

A second guideline for using open source software in education is to build considerable infrastructure to support student learning. This infrastructure may take the form of process as well as a scaffolding of documentation. A development process, perhaps agile, which allows for change during development is necessary for providing structure to project development and student learning. This process should be accompanied by a framework of documentation and communication tools to provide guidance to students.

Due to the flexible nature of open source software and the very changeable nature of humanitarian software, student projects should be well bounded and requirements clearly prioritized. This allows the changes that inevitably occur to be evaluated and adjustments in development to be more easily made throughout the semester.

If the educational experience involves students with varying backgrounds, differing metrics and grading approaches should be planned for the range of students expected in the course. In addition, students may be fulfilling vastly different roles in a contribution to a project and a set of metrics for each of these roles should be established.

Lastly, the creation of a community for development is critical when using open source software as a foundation for software engineering education. The early establishment of a group of professionals, users, and students provides students with guidance throughout development and a community to which to ask questions, as well as serving as role models.

5. Conclusion and Future Directions

Our experience in using the Sahana project as a base for software engineering education has shown that a real-world open source project can successfully support a range of software engineering learning. We have observed substantial benefits to students including very high motivation, hands-on experience with multiple aspects of software engineering, professional growth, and contacts gained by students through interacting with the Sahana community.

In the future, we plan to continue the use of Sahana and other open source humanitarian software as a foundation for software engineering education. A "Topics in Application Programming" course is being offered in spring 2007 which will use humanitarian open source software as a starting point for development. The course has an enrollment of 14 students at Trinity College and an additional four to five students at both Wesleyan University and Connecticut College. In addition, a summer humanitarian open-source software development institute is planned for summer 2007. Six to eight students from the three colleges will participate in team projects under the supervision of a combination of faculty, industry representatives and social service participants.

6. References

[1] Allen, E., R. Cartwright and C. Reis, "Production programming in the classroom", SIGCSE 2003, pp, 89-93.

[2] Apikul, C., "Managing Disasters – Sahana", The International Open Source Network, retrieved from http://www.iosn.net/foss/humanitarian/projects/sahana/ on January 11, 2007.

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

[4] Ellis, H.J.C., Morelli, R.A., de Lanerolle, T., Damon, J., and J. Raye, "Can Humanitarian Open-Source Software Development Draw New Students to CS?", SIGCSE 2007, 2007.

[5] Fernandez, J.D., M. Garcia, D. Camacho, and A. Evans, "Software engineering industry experience: the key to success", Journal of Computing Sciences in Colleges, Vol. 21 , No. 4, 2006, pp. 230-236.

[6] Hawthorne, M.J. and D.E. Perry, "Software engineering education in the era of outsourcing, distributed development, and open source software: Challenges and opportunities", Proceedings of the 27th international conference on Software engineering, 2005, pp. 643-644.

[7] Hislop, G.W., "Scaffolding Student Work in Capstone Design Courses", 36th Annual ASEE/IEEE Frontiers in Education Conference, 2006, pp. T1E 1-4.

[8] IEEE-SA Standards Board, "IEEE Recommended Practice for Software Requirements Specifications", New York: The Institute of Electrical and Electronics Engineers, Inc. 1998.

[9] IEEE-SA Standards Board, " IEEE Recommended Practice for Software Design Descriptions", New York: The Institute of Electrical and Electronics Engineers, Inc. 1998.

[10] Liu, C., "Enriching software engineering courses with service-learning projects and the open-source approach", Proceedings of the 27th international conference on Software engineering, 2005, pp. 613–614.

[11] McGrath, O., "Balancing act: community and local requirements in an open source development process", Proceedings of the 34th annual ACM SIGUCCS conference on User services, 2006, pp. 240–244.

[12] Patterson, D., President's Letter, CACM, Vol. 49, No. 3, 2006, pp. 27-30.

[13] Samuelson, P., "IBM's pragmatic embrace of open source," CACM, Vol. 49, No. 10, 2006, pp. 21 – 25.

[14] Shockey, K. and P. Cabrera., "Using open source to enhance learning", Proc. of 6th ITHET, 2005, pp. 7-12.

[15] Spinellis, D., "Open Source and Professional Advancement", IEEE Software, Vol. 23, No. 5, 2006, p. 70.

[16] Toth, K., “Experiences with open source software engineering tools”, IEEE Software, vol. 23, no. 6, 2006, pp. 44-52.