Innovate: Journal of Online Education

Volume 3
Issue 1 *October/November 2006*

Article 7

11-1-2006

From, By, and For the OSSD: Software Engineering Education Using an Open Source Software Approach

Kun Huang

Yifei Dong

Xun Ge

From, By, and For the OSSD: Software Engineering Education Using an Open Source Software Approach

From, By, and For the OSSD: Software Engineering Education Using an Open Source Software Approach

by Kun Huang, Yifei Dong, and Xun Ge

In the discipline of computer science, the disconnect between professional practice and educational preparation often poses a challenge for academic programs. On the one hand, computing is a complex, multidisciplinary field that integrates knowledge from many different areas. This field calls for a workforce of computer engineers and scientists who are able to work in teams in order to apply knowledge and skills from different disciplines in solving problems, to deal with ever-changing situations, and to update knowledge and skills continuously through self-directed learning (McGettrick et al. 2004). On the other hand, most computing education curricula are organized around compartmentalized courses in which students acquire largely incremental, disconnected knowledge and skill sets. Moreover, many computing courses focus more on end products—that is, on the delivery of workable programs—than on the development processes themselves (Upchurch and Sims-Knight 1997). In contrast to the frequent teamwork found in the computing industry, students usually program individually or in small teams at best (Knight, Prey, and Wulf 1997). Consequently, when faced with complex, real-world tasks, students often find it difficult to synthesize and apply their knowledge and skills to solve problems.

Such a focus on self-contained knowledge, individual work, and final products in computing education reflects an epistemological view that "fosters selective inattention to practical competence and professional artistry" (Schon 1983, vii). This paradigm directly contrasts software engineering practice, which is a process of constantly dealing with situated, ill-structured problems and generating viable solutions through individuals' cognitive and metacognitive thinking (Brown and Duguid 1991). Current educational practice lacks just such dynamic knowledge built *in situ*. We maintain that the software engineering profession involves more than software programming and information technologies. It is both an "information ecology" that consists of people, practices, values, and technologies (Nardi and O' Day 1999, 49) as well as a community of practice that people join through legitimate peripheral participation and build their identities through meaning negotiation and interactions with one another (Lave and Wenger 1991; Wenger 1998).

In searching for ways to instill the sense of community through which students learn by solving complex, ill-structured problems collaboratively, we began to examine a special type of software development community—the open source software development (OSSD) community. The OSSD community consists of a group of dedicated, professional developers and user participants who volunteer their expertise and energy in developing open source software over a long period of time toward a common goal. By utilizing online work space and communication tools, these geographically distributed participants collaborate to solve problems through continuous meaning negotiation and knowledge sharing. We believe that the OSSD approach has the potential to address the educational problems discussed above.

Taking a design-based research approach, we have been conducting a series of two studies at the University of Oklahoma (OU), with one study leading into the other. In the fall of 2005, we studied an open source project called Gallery and analyzed why the project was successful from the perspective of collaborative problem-solving, decision-making, and knowledge construction. Based on our initial findings from this study, we then designed and implemented an OSSD environment in a graduate course on software engineering in the spring semester of 2006; during the implementation of the course, we also began to investigate the effects of the OSSD environment on students' motivation and collaborative problem-solving. This article presents the preliminary findings of the first study on an OSSD community, reports the ongoing study on the effects of OSSD on students, and draws useful implications from the two studies. We hope that this research will not only benefit computing education but also education in other disciplines.

From the OSSD: Characteristics Identified

Significant open source products like Apache, Mozilla Firefox, and Linux demonstrate the marked success of the OSSD community. What characterizes an OSSD community is the dedication of time and energy by a group of volunteers in developing software that is available to use, modify, and redistribute (Hars and Ou 2001). We believe that some OSSD features may be beneficial to current methods of computing education. We therefore selected Gallery, an OSSD community whose size is comparable to that a typical computing class, as the focus of our study. Through careful analysis of the data hosted on the Gallery Web site and other related literature, we have identified several salient characteristics below that contribute to the success of the OSSD community (Ge, Dong, and Huang 2006).

Projects as Shared Enterprise

In an OSSD community, the members' expertise and activities are all geared toward the development of the OSSD project. In other words, the OSSD project drives members' collaboration, knowledge sharing, and learning. If we regard Gallery developers as learners, the starting point of their learning is the Gallery project, and their knowledge and skills are obtained in the process of collaboratively solving complex, real-world problems. Compared with the skills acquired from isolated computing classes, the skills gained from the OSSD development are more holistic and readily applicable to new situations.

Motivation in Joining the OSSD Community

Personal interest motivates OSSD members to join a particular OSSD project. The interest can be in the software program itself or it can arise from the potential of the program to improve the technical skills or marketability of its members (Hars and Ou 2001).

Distributed Expertise and Expanded Community

The knowledge and skills possessed by the OSSD developers are diverse and distributed across the community. Every member has a specialized area. The Gallery team, for example, consists of project managers, developers, database maintainers, graphic designers, user-interface designers, user forum helpers and moderators, Web site maintainers, and translators. The community is not limited to developers; rather, it expands to all kinds of people who contribute to the development of Gallery in various ways. Some may contribute to the source code, others may provide user support, and still others may simply report bugs experienced as users. No matter what expertise the member possesses, everyone can find a legitimate place in the community.

Careful and Systematic Project Planning

Careful and systematic project planning is critical for an OSSD project to ensure the effective collaboration among geographically distributed developers. It gives rise to a roadmap for each of the development goals or milestones, which developers try to reach. In the planning process, the project leader plays an essential role; for example, in the Gallery community, one of the co-founders presents a development roadmap that delineates the project in terms of project features, release schedules, and task distributions. However, the co-founder does not play a dominant role in determining the subsequent steps of the project. Rather, the proposal is subject to the members' scrutiny. Through discussions, the proposal is further defined, clarified, revised, and enriched, ultimately establishing a common ground for members' collaboration.

Collaborative Knowledge Building

OSSD is not only a software development process that leads to a successful and continuously improving product but also a significant learning process for each community member. The Gallery members construct knowledge through multiple rounds of discussions in which they outline problems, share resources and ideas, negotiate solutions, clarify misconceptions, and reach consensus. As a result, each member gains substantial knowledge and experience by working on Gallery.

OSSD System Affordances

In the meantime, the OSSD environment also provides an ideal virtual space that facilitates effective collaboration. The hierarchical organization of the source code provides a clear visual display for the Gallery developers; the task list decomposes the project tasks into different priority levels and allows the developers to take over tasks and report progress. Different mailing lists allow members to join ongoing discussions and keep updated with any new progress of the software. Bug reports, feature requests, and discussion forums help to establish connections and communications to the wider community, which, in turn, enhances the project improvement.

By the OSSD: Design-Based Research

We believe our findings about the OSSD community are promising, and we see the potentials of implementing its features to address some of the concerns generated by more traditional computing courses. For the spring semester of 2006 at OU, we designed and implemented a computing education course—Computer Science 5213: Software Engineering Processes—with a simulated, full-featured, OSSD environment. Concurrently, we also conducted exploratory research on the cognitive and affective dimensions of the OSSD environment in computing education. The course was a graduate-level course on software engineering processes, and a total of 19 students were enrolled during its first semester. We took an ethnographic approach for the study that used a combination of techniques including conducting observations and interviews and examining student artifacts. The implemented OSSD features are explained below.

Group Forming: Two-Way Selection

The group-forming process was adapted from real OSSD practice but also modified to suit the instructional needs of the course. At the beginning of the semester, the students submitted their resumes to the instructor. Based on the students' experiences and skills, the instructor identified four project founders; interviews were then conducted in the classroom between the founders and the rest of the class in the fashion of a job fair. In these interviews the founders assumed the role of recruiters, and the other students assumed the role of job seekers (Exhibit 1). The process of selection was two-way: Based on their rough ideas for projects, founders could review applicants' resumes and talk with them in order to discover the best match for their needs while the students could shop around to apply for those projects that interested them the most. After the interviews, the founders and job seekers submitted their respective lists of "most-wanted" participants or positions to the instructor, and the instructor tried to match the projects with the students in a way that served the best interests of both parties.

There were at least two benefits for forming groups this way. First, the interview increased the students' self-awareness. The applicants had to think about their knowledge and skills as compared to the project needs and identify the holes in their expertise; in turn, the founders had to reflect on what types of skills and experiences their projects required. Secondly, the four different projects provided the students with selection options, thus promoting their autonomy.

Simulated OSSD and Meaningful Projects

Once the groups were formed, all of the group members discussed the details of their projects based on the ideas initiated by the founders, and then they submitted their project proposals to the instructor. The proposals in this case included plans for a document routing and tracking system (DocBUS), a graphical interface for a local area augmentation system (iLAAS), a Web-based smart event scheduler (SchedulePlus), and an online war memorial (Norman World War II Online Memorial). All of the proposed projects were based on real needs or from real sources, which helped ensure the authenticity of the learning activities. Compared with artificial teacher-assigned projects, these projects were more meaningful and relevant, and the student founders felt more ownership and investment in them. In addition, the selection of project founders ensured that there would be at least one experienced member in each potential group; by working together, the group members could then learn from the valuable experiences shared by their more knowledgeable peers.

Real World Projects as Anchors for Learning

The whole course was organized around the development of the software project. Though the course treated software engineering processes, the students did not learn theories and practice in isolation. Rather, the projects served as anchors for students to learn software engineering processes by actually following the process of designing and developing real-world software.

For example, as in the real OSSD environment (e.g., Gallery), the students had to plan and document their design and development carefully for each milestone in the process in order to make team collaboration possible (Exhibit 2). These updates were geared toward further enhancing development as well as measuring progress; consequently, the knowledge gained was dynamic rather than inert (Bransford, Brown, and Cocking 1999). Furthermore, such a process avoided a common problem many computing students have—their tendency to go directly to software design and development without sufficient preliminary planning (Upchurch and Sims-Knight 1997). By working on an actual project designed for specific, real-world application, students were provided with a clear foundation that guided their design and problem-solving activities from one stage to the next.

Open Source Thinking and Collaboration

In this course, open source was not limited to the access and use of source codes; more important were the open source thinking and collaboration processes. Since the students communicated through the OSSD system and saved their project documents, meeting minutes, personal reflections, and major decisions in the system, their thinking and collaboration processes became transparent and open to examination, problem-detection, critique, and suggestions. This emphasis on process in the OSSD environment resulted in three key pedagogical advantages for the course as a whole.

First, the open source processes helped to make students' thinking visible and encouraged elaboration and reflection (Linn, Bell, and Hsi 1998). The resulting metacognitive skills are especially important to ill-structured problem-solving, which the students will encounter in their work on a daily basis.

Second, the open source processes also helped to enhance team collaboration. Given the complexity of the real-world projects, it was essential for team members to collaborate effectively on their team project. Since every member was responsible for a project component, which was also connected to other members' components, it was important for the students to see others' work and progress. In this case, the open source approach facilitated the individual members' contribution to the team and the integration of their work into a whole project.

Third, the transparent and visible open source processes allowed the instructor to perform formative assessments constantly as the students developed their projects. The instructor assessed the team progress and the student performance by examining all the logs stored on the OSSD system, including mailing lists, discussion forums, project documents, task lists, source codes, and their contributors. At the same time, students could also log in to the OSSD system to review the work and progress of other individuals or teams, which made them more aware of their own performance.

Instructor as an Expert, Coach, and Project Manager

The instructor plays an important role in the OSSD learning environment, not only as an expert but also as a coach and a project manager. The instructor provides useful advice and suggestions from the perspective of an expert. The instructor also plays the role of a coach, attending to individual students' needs and facilitating students' learning and collaboration processes (Exhibit 3). As an expert and a coach, the instructor detects the bugs during student learning processes, provides the students with in-time diagnostic feedback, and reorients them in the right direction. In addition, the instructor also serves as a project manager who oversees, monitors, and facilitates all the teams and their projects.

Students as Both Developers and Users

In a real OSSD community, members with different levels of knowledge and experience play different or multiple roles, such as developers, testers, or users. This course is designed to encourage all the students to be both developers and users as they participate on two levels of the learning community—the level of the individual team as well as the level of the class as a whole. Within each team, every member works as a developer on a certain module of the software and serves as a user for the other team members in the software development process (Exhibit 4). Moreover, the teams are also encouraged to follow the extreme programming (XP) model (Beck 2000), which features iterative development and frequent releases. Once a concrete product is released within the class, members of other teams can test it and provide suggestions and feedback as users. The students may download and use the software, detect and report bugs, request additional features, or even fix bugs and provide patches for the other teams. In a traditional course setting, it is usually hard for some students to find ways to contribute, but in this course every member has a place on the team and a role to play within the class as a whole.

In many project-based courses, it is often the case that every team only takes care of its own business and that team members know or care very little about other teams' projects. In this respect, learning in a single context may prevent the transfer of knowledge (Bransford, Brown, and Cocking 1999). However, by comparing their own projects with other teams' projects and by assuming different roles (e.g., developers and users), students are more able to gain substantial understanding of software engineering processes—and, in turn, are better able to transfer their knowledge to new problem-solving situations.

OSSD Virtual Space

The course Web site implemented a similar OSSD virtual working space using GForge, an open source branch of the popular OSSD support platform SourceForge. The support of an OSSD Web site is necessary because of the complexity of the projects and the substantial amount of time required for meetings.

The students were encouraged to utilize the OSSD Web site fully, and their good practice was demonstrated to the class. All four projects were hosted on the same Web site, and each project also had its own home page. The students used this Web site to save and share all their source codes, which were organized into different categories; they also used the Web site to record meeting minutes or major project decisions, plan and allocate to-do tasks, share resources and tools, use mailing lists to communicate with group members, and ask group members or the whole class for suggestions and help (Exhibit 5).

The OSSD Web site thus provided an open record of the student' projects, which allowed students to refer to and discuss their team projects and allowed the instructor to monitor the projects and the team process. The Web site also facilitated student collaboration by allowing them to communicate at any time and to plan, divide, and trace tasks. The use of OSSD tools and the sharing of its resources clearly produce a community knowledge base that is closely tied to the development of software projects.

For the OSSD: Implications

Although our implementation and research are currently in progress, the initial findings and the feedback from the students are promising. Our preliminary data analysis of the Spring 2006 class suggests that students demonstrate a high level of motivation toward their projects, especially in the case of groups that have frequent interactions with real clients. During interviews the students expressed their excitement about having the chance to develop a software project from scratch. In their responses to course evaluation surveys administered at the end of the semester, the students mentioned a number of things that they had learned through working on the real-world projects in the OSSD environment:

- They learned how to find a position and play a role in their teams in order to make unique contributions to a large-scale project.
- They gained knowledge based on project needs, and they learned from more knowledgeable peers.
- They learned to use the OSSD virtual space to manage large-scale projects with which they had little previous experience.
- They felt more capable of working with different people.

Furthermore, we found that factors such as project authenticity level, instructor intervention, and project founders' leadership had a great influence on the success of the classroom OSSD projects. More successful projects such as DocBUS moved beyond the scope of the course as some members continued to work on the project even after the course was finished.

As our design-based research continues, we hope to identify special characteristics of OSSD in the classroom setting and to gain a deeper understanding of student cognition, motivation, and behavior in the OSSD environment. We also hope to generate instructional strategies for implementing successful OSSD in the classroom setting. In the future we look forward to building a project database and forming partnerships with other computing programs, universities, industries, or companies. In addition, we would also like to extend the OSSD concept and approach to other disciplines such as math, science, and engineering.

Conclusion

We hope that the OSSD approach provides an alternative instructional approach to address the issues and challenges faced in computing education. Ideally, computing education nurtures graduates who are competent in domain knowledge, skilled in complex problem-solving, and capable of self-directed learning. We expect that our future research will shed more light on the solution, so that what we as educators learn from the OSSD approach—both the process and the product—will allow us to contribute back to a larger computing community by training highly qualified software engineers.

References

Beck, K. 2000. *Extreme programming explained*. Boston: Addison-Wesley.

Bransford, J. D., A. L. Brown, and R. R. Cocking. 1999. *How people learn: Brain, mind, experience, and school*. Washington, DC: National Academy.

Brown, J. S., and P. Duguid. 1991. Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organization Science* 2:40-57.

Ge, X., Y. Dong, and K. Huang. 2006. Shared knowledge construction process in an open source software development community: An investigation of the Gallery Community. Paper presented at the 7th International Conference of Learning Sciences, Bloomington, IN, June-July.

Hars, A., and S. Ou. 2001. Working for free?—motivations of participating in open source projects. Paper presented at the 34th Hawaii International Conference on System Sciences, Outrigger Wailea Resort, Hawaii, January. http://csdl2.computer.org/comp/proceedings/hicss/2001/0981/07/09817014.pdf (accessed September 30, 2006).

Knight, J. C., J. C. Prey, and W. A. Wulf. 1997. Undergraduate computer science education: A new curriculum philosophy & overview. Paper presented at the 1997 Frontiers in Education Conference, Pittsburgh, PA, November. http://www2.umassd.edu/swpi/UVa/curriculum.pdf (accessed September 30, 2006).

Lave, J., and E. Wenger. 1991. *Situated learning: legitimate peripheral participation*. Cambridge: Cambridge University Press.

Linn, M. C., P. Bell, and S. Hsi. 1998. Using the Internet to enhance student understanding of science: The knowledge integration environment. *Interactive Learning Environments* 6 (1/2): 4-38.

McGettrick, A., R. Boyle, R. Ibbett, J. Lloyd, G. Lovegrove, and K. Mander. 2004. *Grand challenges in computing education*. Swindon, UK: The British Computer Society.

Nardi, B. A., and V. L. O'Day. 1999. *Information ecologies: Using technology with heart*. Cambridge, MA: The MIT Press.

Schon, D. A. 1983. *The reflective practitioner*. New York: Basic Books, Inc.

Upchurch, R. L., and J. E. Sims-Knight. 1997. Designing process-based software curriculum. Paper presented at the Tenth Conference on Software Education and Training, Virginia Beach, VA.

Wenger, E. 1998. *Communities of practice: learning, meaning, and identity*. Cambridge, MA: Cambridge University Press.

COPYRIGHT AND CITATION INFORMATION FOR THIS ARTICLE