An Investigation of an Open-Source Software Development Environment in a Software Engineering Course

Xun Ge, Kun Huang, and Yifei Dong

Abstract

A semester-long ethnography study was carried out to investigate project-based learning in a graduate software engineering course through the implementation of an Open-Source Software Development (OSSD) learning environment, which featured authentic projects, learning community, cognitive apprenticeship, and technology affordances. The study revealed that while the OSSD learning environment motivated students to engage in real-world projects, tensions arose between the the innovative learning environment and the students' self-processes—their perceptions, expectations, beliefs, goals, and values. Most importantly, this study demonstrated key interplays between project authenticity and learner characteristics, which resulted in different identity representations and different perceptions among students, which in turn affected students' goal orientations, motivation to work on projects, commitment to team collaboration, attitudes toward expert coaching and feedback, and the use of collaborative technologies.

Introduction

It has been accepted by many educational researchers that learning and cognition are situated in the physical and social context in which learning takes place (Brown, Collins, & Duguid, 1989). Yet, it is widely acknowledged that in our educational system, the culture of schools is often disconnected from the culture of practitioner communities (Resnick, 1987). Education in computer science is no exception. Computer science curricula are often organized around capsulated courses, which do not reflect the complexity and ill-structuredness of the real-world experience (e.g., Carrington & Kim, 2003; C. Liu, 2005). Knowledge acquired by students is often inert (Bransford, Brown, & Cocking, 1999). When

faced with complex, holistic, real-world projects, students have difficulty synthesizing, integrating, and adaptively applying knowledge to solve emerging problems (Feltovich, Spiro, Coulson, & Feltovich, 1996). There is a gap between the current state of computer science education and the expectations of the software industry, and our current education is not adequately preparing students for real-world challenges. This situation poses great concerns and challenges to computer science educators.

In an effort to address the problems described above, many educational researchers proposed experiential instructional strategies such as project-based learning and problem-based learning (PBL) to engage learners in authentic, complex problem-solving activities (Barrows & Tamblyn, 1980; Blumenfeld, et al., 1991; Hmelo-Silver, 2004). While project-based learning is understood differently by different researchers, Krajcik and Blumenfeld (2006) defined five key features representing a project-based learning environment: 1) students start with a driving question or problem; 2) students participate in authentic inquiry processes and apply domain knowledge to solve problems; 3) students, teachers, and community members work collaboratively in the problem solving process; 4) technology tools are often used to scaffold the problem solving activities; 5) students create tangible products that address the driving question or problem. These features of project-based learning bear many similarities to PBL, but in PBL the learning experience is often centered around "the investigation, explanation, and resolution of meaningful problems" (Hmelo-Silver, 2004, p. 236), which may or may not involve creating tangible artifacts. In addition, Savery (2006) argued that in project-based learning, expected outcomes are clearly defined and consequently learners have less freedom in defining the outcomes and goals of a project. Despite the differences, both project-based learning and PBL afford learning tasks that resemble real-world experience, and as a result, knowledge becomes in part "a product of the activity, context, and culture in which it is developed and used" (Brown et al., 1989, p. 32). The implementation of these two types of learning environments has witnessed some success in improving problem-solving skills and outcomes (e.g., Boaier, 2002; Cognition and Technology Group at Vanderbilt, 1992; Jeong & Hmelo-Silver, 2010; Vernon & Blake, 1993), as well as in promoting learners' positive perceptions of these learning environments (e.g., Azer, 2009; Denton, Adams, Blatt, & Lorish, 2000; M. Liu & Rutledge, 1997).

In their discussions on architecting learning environments, Barab and Duffy (2000) classified PBL as one type of "practice fields" (p. 31), which was defined as instructional contexts that engage students in problem-solving activities they are likely to encounter in real life. Barab and Duffy (2000) distinguished *practice fields* from the concept of *communities of practice*. While acknowledging the benefits of practice fields in addressing the issue of decontextualized learning, Barab and Duffy (2000) argued that practice fields do not equal communities of practice in that the former is still separate from the real world;

and that the activities learners engaged in are still school tasks abstracted from the real practicing community. A community of practice is different from a practice field in that it involves a group of individuals participating in community activities, engaging in and contributing to the practices of the community, and continuously developing shared identity in the community (Lave & Wenger, 1991). The distinction between the practice fields and the communities of practice is important because the two environments may have different impacts on learners' motivation, self-perceptions, perceived values and utilities (Eccles et al., 1983; Maehr, 1984; Wigfield & Eccles, 2000).

The Open-Source Software Development (OSSD) community represents a good example of communities of practice. Open-source software is based on the openness of software source code which allows the access, modification, and redistribution of software products (Raymond, 1998). In an OSSD community, software developers and users work together on improving a software product as a virtual community (Raymond, 1998). A certain online collaborative platform (e.g., SourceForge, GForge, CollabNet, and Google Code) is normally used by the community members to store software source code, exchange ideas, plan and delegate tasks, interact with users, and respond to bug reports and feature requests. The close collaboration between software developers and users, supported by an online collaboration space, allows software developers to quickly respond to users' needs and requests and subsequently modify and improve the software to satisfy the users' needs (Raymond, 1998). The OSSD community members often volunteer their time, effort, and expertise for the common goal of developing user-friendly, quality software products for the good of the public. In the mean time, OSSD community members also benefit from their active participation by accumulating experiences and skills (Hars & Ou, 2001).

Intrigued by some successful OSSD communities (e.g., Mozilla Firefox, Linux), we started to explore ways to implement such a community in computer science education. To this end, we studied a real-world, successful OSSD community (see Ge, Dong, & Huang, 2006 for details). Based on our findings, we implemented an OSSD learning environment in a graduate course of software engineering as an innovative approach to teach software development processes. The OSSD learning environment was characterized by the following features: authentic projects, real clients and users, students' engagement in authentic inquiry processes, a community of practice (including students, professor, users, and clients), collaboration tools (including a virtual collaborative working space), and the tangible artifacts of software products.

The OSSD learning environment reflected the features of both project-based learning and PBL, but most importantly it resembled a real OSSD community. It was our expectation that these features of communities of practice would enculturate students into a community of software engineers. At the same time, we also recognized the constraints of time (e.g., limited by a semester), classroom settings, and the curriculum influencing the implementation of the OSSD community. Therefore, the OSSD learning environment was not

strictly a community of practice but an intentional move away from practice fields toward communities of practice (Barab & Duffy, 2000). As such, the implementation of the OSSD learning environment provided us with an opportunity to examine the OSSD culture and the interplays of various components afforded by this particular learning environment.

To date, few studies have empirically investigated the culture, students' perceptions, motivation, and experience in a learning environment situated between *communities of practice* and *practice fields*. Among the existing few studies that are similar to this study, Barab, Barnett, and Squire (2002) investigated a teacher education community consisting of both in-service and pre-service teachers, from which they found the following tensions experienced by the participants: 1) instructor as a facilitator versus a gatekeeper, 2) learning theory versus doing practice in this community, 3) student-submitted portfolio as support for reflection versus an accountability device, and 4) stability versus change in the community. Another two-week study examined the cultural shift taking place in a high school computer class that was transitioned from a traditional learning environment to an open-ended learning environment bearing some community features. From the study, Ge, Thomas, and Greene (2006) observed some intransigent cultural features interfering with the culture in the new learning environment. Although carried out in different contexts, both studies invariably indicated the benefits for learners as they participated in the learning community, as well as issues and challenges encountered in the process of implementing communities of practice in the classroom settings. We were motivated by these studies to further explore those complex issues in our unique OSSD learning environment and to uncover various factors that created tensions and influenced learners' perceptions and experiences as they were immersed in the culture of a community of practice. It is expected that the findings from this study will shed light on the instructional design and implementation of similar learning environments. Thus, the purpose of the study was to investigate the following questions:

1. What is the OSSD culture like? How do the students coming from the traditional learning environment react to the new OSSD culture?
2. How does the OSSD environment interplay with students' self-processes (e.g., perceptions, prior knowledge and experience, goals, and motivation)?

Method

Participants and the OSSD Learning Environment

The participants were nineteen graduate students (16 male, 3 female) enrolled in a software engineering processes course in a major southwestern university. The class met twice per week, for 75 minutes each, over a semester of fifteen weeks. The class was designed to

simulate the professional activities and problem-solving processes found in a real OSSD community. At the beginning of the semester, four students who had more prior working experience were identified by the professor as project founders. The responsibilities of the project founders were to propose a software project and conduct in-class interviews to recruit team members from among their classmates to work on their proposed project. After the interviews, based on the preference expressed by the project founders and the rest of the students, the professor assigned the students into four project teams. The four projects were Doc (a document routing and tracking system), Museum (an online WWII memorial), Landing (a local area augmentation system used for flight landing), and Scheduler (a web-based event scheduler). The project names are pseudonyms in order to protect the participants' privacy. The rest of this article will refer to each of the teams by the pseudonym of the team's project.

As soon as the teams were formed, they began to work on their proposed software project, which lasted till the end of the semester. The aim was to develop software products to serve the needs of identified users. A typical class was characterized by the professor introducing some important concepts on software engineering processes, followed by class discussions. During the class discussions, students were often asked how the concepts were related or could be applied to their projects. The teams met weekly outside the class to discuss project progress, tackle technical difficulties, plan for next steps, and divide up tasks. A substantial amount of the team project was completed by individual members outside the class. Throughout the semester, each team had to make several presentations to the class about their project, especially when a new version of the software was released. The presentations were followed by class discussions. As part of the course requirement, students were asked to write weekly reports to journal what they had learned and done in the past week, their project progress, as well as their plan for the upcoming week. By the end of the semester, each team had gone through several milestones of software development and released several versions of their software product.

A web-based OSSD project management system, GForge, was used to host all the projects and served as a virtual platform for team collaboration. The system had a home page which listed all four projects, each of which was hyperlinked to its own home page and a set of tools and resources. All the information in the system was accessible to all the students. The students were required to use the GForge system for their project but there was no specific requirement as to which tools the students had to use. The professor provided scaffolding through lectures and meetings (team or individual) whenever necessary, to supply relevant information, domain knowledge, and useful resources to students. Throughout the semester, the professor actively monitored and facilitated the progress of each team's project.

The students were informed from the first class that they would be working throughout the semester on their team project. The students were further told that there would be no exams or quizzes; instead they would be evaluated by their final projects as well as their performance and progress throughout the semester. The professor emphasized in class several times during the semester that grades were not the emphasis and purpose of the class; instead, experiencing the software engineering processes by working on their project was the goal of the class.

Design

Since the main purpose of this study was to understand, describe and interpret the cultural and social system situated in the OSSD environment, the ethnography approach, characterized by field observations and interviews, was used to examine the patterns of student behaviors, language, interactions, attitudes and feelings in the culture of the OSSD learning environment (Creswell, 1998; Spradley, 1980).

The research team consisted of three members, a professor of instructional psychology and technology (IPT), a professor of computer science, and a graduate assistant (IPT doctoral student). Each of the 29 class sessions was observed by at least one researcher and field notes were taken by following a detailed observation protocol. Reflection notes were written immediately after the observation, which were shared and discussed at the weekly research team meetings. Toward the end of the semester, eighteen students participated in a semi-structured interview, which lasted between 30 to 60 minutes. All the interviews were conducted by the IPT professor and the graduate assistant. The interview data were kept away from the computer science professor until the semester was ended and grading was completed. The interview questions were generated based on observations and other data sources (e.g., students' weekly reports and artifacts). Specifically, the interviews asked students' to describe their background information and their general impressions of the class, as well as their specific experiences in the OSSD learning environment, including their projects, team collaboration, the professor's role, and the virtual collaboration system.

Researchers' Roles

The computer science professor had over 20 years of real-world experience in software engineering and had taken leading roles in several commercial and academic software projects. He played dual roles in this study. He was the instructor of the software engineering course, playing the role of a facilitator, coach, and domain expert. At the same time, he was also a participant observer and researcher. Due to his instructor's role, he was not involved in conducting interviews. The students were made clear at the time of the interview that the professor had no access to the interview data while the semester was in session so that they could be assured to speak freely with the other two researchers.

The graduate assistant had extensive experience in a couple of qualitative research projects in the past, and so she was skilled in participant observations and ethnographic interviews. She attended all the weekly classes in order to record field notes. In addition, she also joined the students' team meetings outside class as frequently as possible. Being a student herself, she was able to develop a good rapport with the students. As a result, she was well informed about the students' activities, project progress, and peer interactions. Therefore, she was another participant observer in the research team, but she was more involved in the data collection process than the professor of computer science.

The IPT professor had years of experience in conducting qualitative research. Therefore, she coordinated the research team's effort and monitored the progress of this research project. She observed some of the classes, particularly on days when there were some important instructional events, for instance, when students presented their new releases of software. At all times, she followed the students' project development by checking students' activities online and attending weekly research meetings. By comparison, she was not as immersed in the field as the other two researchers. This kind of research identity had enabled her to examine and interpret data from a relatively distant perspective.

Although the two IPT researchers did not have domain knowledge in computer science or software engineering, they were able to complement the computer science professor's expertise by contributing their knowledge and skills in instructional design and conducting educational research. Their expertise allowed them to examine the OSSD learning environment from a perspective different from that of the computer science professor. On the other hand, varying degrees of involvement in data collection and the different roles played by the three researchers (Spradley, 1980) had created a nice balance between *emic* (experience-near) and *etic* (experience-distant) perspectives (Geertz, 1976; 1983). These different roles and perspectives enabled the triangulation of data interpretation and inferences, which helped to validate findings and conclusions. The researchers acknowledged their influence on the environment under investigation.

Data Sources and Analysis

Twenty-nine observation notes and eighteen interview transcripts were the major sources of data. In addition, students' weekly reports and their project files, including source code and project documentation archived in the GForge system, were also collected as supplemental data sources.

Data analysis was conducted hand-in-hand with the data collection process. The research team met weekly to discuss, interpret, and negotiate the meaning of the most salient data collected each week, including observation and weekly reports, teaching and learning activities, team collaboration, project progress, and any emerging concerns. As a result of the collective coding by the research team, initial coding categories were generated and subsequently applied to code the observation notes. Later, patterns and

themes were identified during the coding process. As new data were gathered, initial themes were confirmed, modified, or eliminated; meanwhile, some new emerging themes were identified.

The interview transcripts were openly coded first by the IPT professor and the graduate assistant. Then, the codes were organized into different categories (e.g., project, professor, team collaboration, and the virtual collaboration system). Within each category, themes were identified, which were then triangulated with the themes emerged from the observation data. The triangulation helped to confirm and complement the themes generated. It also helped the researchers to link the observed behaviors to the internal processes of the participants. Consequently, relations among the themes were mapped and a logical chain of evidence was established (Miles & Huberman, 1994).

Findings

General OSSD Culture

Overall, the OSSD environment was characterized by a culture featuring identity development and a sense of connection with the larger software engineering community through working with real projects and clients, collaborative team work, and cognitive apprenticeship from the professor through mentoring, coaching, and scaffolding. However, we also found some misalignments between the students' perceptions and the goals of the OSSD environment, as well as contextual constraints in the implementation process, which are illustrated below.

According to our field notes, most students felt really excited about their developer's role, and they tried to live up to that role by acting like professional software developers. Observation and interview data suggested that most students carefully prepared themselves for the simulated interviews at the beginning of the semester. On the day of the interviews, they tried their best to present themselves to the founders/leaders of the teams, whose projects they were interested in. Meanwhile, the founders of the teams carefully reviewed the members' résumés before the interviews so that they could ask relevant questions in order to recruit the most suitable members for their projects. We noticed that those simulated interviews were close mimics of job interviews experienced by professional software engineers in the real world. As the teams were assembled and the projects unfolded, the students had meetings with their clients and began to work on user requirements based on the clients' feedback. Throughout the process, the professor played the role of an expert providing cognitive apprenticeship to the students.

The authentic nature of the OSSD environment stimulated the students' interest and kept them persistent in their pursuit of the projects. For example, by the end of the semester, the members of the Doc team planned to obtain a license for their software,

while three members of the Landing team planned to continue working on their project after the semester ended. At least half of students indicated in the interview that they did not develop software to satisfy the course requirements per se, but also to satisfy the needs of their clients.

While a majority of students understood the purpose and the meaning of the OSSD environment and took advantage of the benefits afforded by this environment, not every individual could embrace the new learning culture. Some students, like Aaron, the leader of the Museum team, felt uncomfortable about the fact that they did not have quizzes or assignments like they usually received in other courses, but rather they had to work on the project for the entire semester without some kind of paper-pencil test. Obviously, these students had not thought that artifacts such as weekly reports, project progress updates with the professor, and feedback from the class, were some forms of formative assessment.

Some students were concerned about the fact that they would have to complete a real-world project within a course, which seemed almost impossible to them. For example, Sandy from the Museum team stated, "The professor-assigned projects are doable within time limit, but client-requested projects may be impossible." Tina from the Scheduler team preferred projects assigned by the professor because they were "all laid out," whereas in this course the projects were "not as structured as other courses." Tina wanted to know exactly what the professor's expectations and requirements were, from which we inferred that she would do just enough to get by the course requirements. It seemed that the complexity and the open-endedness of the OSSD learning environment were in conflict with some students' perceptions and beliefs about learning and instruction, which were rooted in their past experience with the conventional, well-structured courses and learning activities. Such a conflict gave rise to the students' sense of discomfort and insecurity in the OSSD learning environment.

In addition, we also noticed that being situated in the school system and classroom context, the OSSD learning environment was distinctively different from the OSSD community of practice. As indicated by the professor and some students, the semester structure made it difficult to have sufficient time to complete a project at a satisfactory level. The students had to narrow down their project scope or give up some efforts that could have led to better solutions and functionalities had they been given more time.

Interplays between the OSSD Environment and the Students' Self-Processes

While the findings presented above provided an overall picture of the OSSD culture, the data revealed more specific details about the interplays between the major components of the OSSD environment (the projects, the learning community, the expert guidance and feedback, and the online collaboration system) and the students' self-processes, including perceptions, attitudes, prior knowledge, and goals. The major themes are reported below in the hope of uncovering the complexities in the OSSD learning environment.

Project Authenticity

The authenticity level of the projects appeared as a determining factor in the OSSD environment. Although the projects were authentic in nature, and all the students worked in the same learning environment, it appeared that project characteristics (e.g., different degrees of authenticity, with or without clients) interacting with learner characteristics (e.g., different prior knowledge, beliefs, goals, and values) led to the differences in the students' perceptions of their team projects and their identity in the context of the projects. Consequently, we found a continuum of identity, ranging from those who identified with software developers and actively engaged in providing solutions for customers, to those who perceived themselves as students working for a grade. The findings below illustrate the different levels of project authenticity.

While all the four team projects were developed for real users, there were different degrees of authenticity varying from having real and specific clients to having a general group of users whose needs or interest were not specified. For example, the Doc and the Landing projects were introduced to the teams by their respective team leaders from their workplace, where they worked as full- or part-time employees. The goal of these two projects was to improve certain areas of performance in their respective organizations. By comparison, the Scheduler project was initiated by its leader based on his own life experience, which was to create a smart scheduler for public users (i.e., with no particular group of users). The Museum team's project was requested by a World War II veteran to establish an online memorial for a WWII naval base. Although the project was authentic because it was requested by a real client, the users of the product were not explicitly specified since users could be anyone from WWII veterans to the general public.

The varying degrees of project authenticity appeared to affect the extent of clarity and explicitness of the projects in terms of tasks, goals, problems, and paths to solutions. This was evidenced in some simulated interviews that occurred at the beginning of the semester. For example, Chris, the leader of the Landing team, had a pretty clear initial idea about their projects: what needed to be accomplished, what components must be included, and what skills were required in order to develop the software. During the simulated interviews, Chris showed his interviewees a printed copy of the project prototype. He explained the project to them with details, and additionally he asked them specific, technical questions such as, "What programming languages do you use?" By comparison, the interviews conducted by Mark, the leader of the Scheduler team, were rather casual. The leader did not seem very well prepared in terms of the specific requirements of the project. Mark recalled in an interview later:

> Um . . . it scared me at first. The idea was kind of daunting . . . one of the things I felt myself when I wrote in my analysis of the interview or whatever was that, I felt myself not having a full, not having the entire requirement analysis, I wasn't

> sure what we were going to do completely . . . I have an idea of what I want to go with that project, but I have not developed a full idea, and I was not able to do what have mounted to a full understanding of whom I needed.

The level of project authenticity was also reflected through the teams' interactions with their clients. For instance, the Doc team had been able to continuously interact with and obtain feedback from their client, that is, the team leader Gajra's workplace. Shortly after the first release of the software, the team visited the organization, met with their client, and presented the release directly to them. The client's feedback and their acknowledgement of the students' work were motivating to the team. As a Doc team member, Gary remarked his experience in the interview, "It was kind of a formal business meeting. Big table, people . . . it was just the time to realize that we are not working in the air." In comparison, the Scheduler team were not able to receive client's feedback because they had a hard time identifying a specific client or user group. Their user analysis and the specifications of the project were mostly based on individual members' common intuition and their perceived needs from their personal experience. Overall, although the presence of a client does not constitute all about authenticity, it is an important aspect of authenticity in this particular OSSD learning context.

Students' Perceptions of the Projects

The authenticity levels of the project, interacting with students' prior experiences, values, expectations, goals, and beliefs, appeared to affect the students' perceptions of the project, which varied from the view that "This is mainly a class project for a grade" to the view that "The project is a beneficial learning opportunity." Along this continuum, our data analysis has identified four major types of student perceptions about their project (see figure 1). For each type of perception, we provide a representative case, supported by data and illustrated with our interpretations of the data.

Case 1. Tina – Scheduler team member. Tina perceived the project mainly as fulfilling a class requirement in order to obtain a grade. Her attitude could be partially attributed to not having a specific client. With that perception, she expected the professor to specify all the requirements for the project so that her team would not need to work hard on defining the project requirements and specifications.

In comparing this course with the past software engineering courses she had taken, Tina commented:

> Compared with other [software engineering courses] this one isn't as structured . . . but in [other] software engineering [courses], it was all laid out . . . I'm better with structured. Because if I don't have to [do it], I'm not gonna do it, you know . . .

From this quote, it was obvious that Tina preferred an "all laid-out" structure for the course. It was inferred that she would like to have the professor tell her what she needed to work on, and she would do just those required by the professor instead of putting extra effort beyond the requirements. With this mindset, Tina was not interested in other teams' projects (as indicated in her interview and confirmed by our observations). In addition, Tina did not appreciate other teams' feedback on her team's project, as indicated by her statement:

> I guess like, I wouldn't want anybody pointing out flaws in my program that I have to fix . . . If it's not good, I'm still going to get an A. You know what I mean? So it doesn't matter so much . . .

Tina's statement indicated that if she had done enough to get an A, she would not bother doing extra work. However, when asked whether she would like others' feedback if it were a real project at work, Tina replied, "I would, in that setting, because you would actually, I mean you would want to be perfect . . ." Therefore, it was inferred that a project without a specific client would not be perceived high on the authenticity continuum by some students. The authenticity level of a project could be a factor potentially influencing a student's perception of a project and his or her decision on how much effort he or she is willing to commit. In Tina's case, she obviously did not treat the Scheduler as a real project. Her past experiences from other courses, together with her perceptions about the authenticity of this project, could contribute to her attitude that this project was merely a required course assignment that she had to work on to earn a grade.

Case 2. Mark – Scheduler team leader. In the same Scheduler team, Mark, the leader of the team, exhibited a different perception from Tina's. While agreeing that the project "definitely has a class feel . . . not expansive enough to be a practical problem or practical product," Mark nonetheless considered the project as an opportunity to learn new skills and experiment with new technology:

> Yes, actually I do [feel motivated to work on the project], and it's not necessarily because of the class, it's because of some of the components that we are putting in . . . are leading-edge web-based technology, things that will be emerging and stuff. That is, that is will-be applications, so that in itself is a motivation for me. It's not necessarily the class-based work, but more about the being able to have a practical . . . environment to work on this sort of experimental new technologies.

Besides, Mark also used the project to practice his project management skills. He said:

> On the first day of class [the professor] asked us our reason for taking this class. One of the reasons I said was I hoped to go into project management,

> software engineering project management, and all this is going to include assembling teams, and developing requirement analysis, and developing basic, or developing from very abstract idea to a very concrete plan for developing a product. So yeah, I think it's pretty beneficial to what I'll be doing, or what I hope I will be doing.

Although the Scheduler project was relatively low in authenticity, Mark perceived the project as a chance to learn new skills and prepare himself for future careers. Thus, he positioned himself as both a learner and a future software engineering project manager. Mark played an active role in his team, taking upon himself a large portion of the project while coordinating his team's collaboration efforts.

Case 3. The Doc team and the Landing team. As mentioned earlier, the Doc team and the Landing team both had specific organizational clients and users, which helped the team members to experience the real-world nature of the projects. Habib, a Doc team member noted:

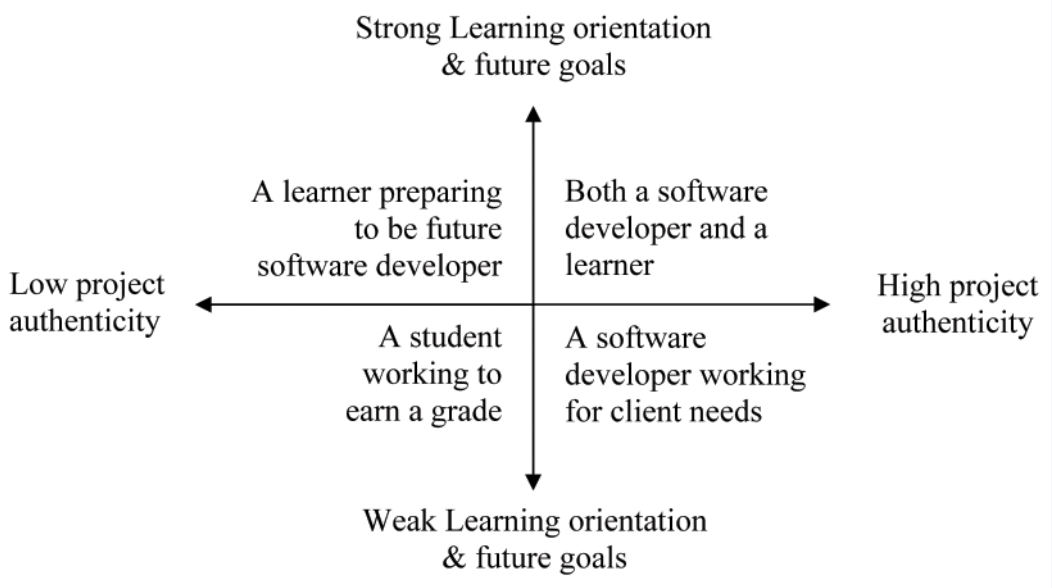
> It was almost like a real world project, because someone is going to use it after we complete it . . . in other courses all you do is they give you some task, you do, you test, you run. But this was development and stuff. They are going to use it in the [client organization's name]. It goes to a real user. We did a presentation for them and they gave us the feedback saying that this has to relate to this. I never had that kind of experience before. It was new.

Comments like the one made by Habib were constantly heard in the interviews with the members from the Doc and the Landing teams. For example, Pavi from the Landing team specifically stated that, "It is not only for course credit . . . it is a real project and it is going to be used . . ." These students were more inclined to identifying themselves with software engineering professionals and more driven by the goal of meeting the clients' needs. Therefore, they were seen as more willing to put in efforts to constantly improve the quality of their software products. Compared with the other two teams, the members of these two teams mentioned their clients much more frequently in various occasions, such as in class presentations and interviews. They demonstrated a clear understanding of the current problems in their client organizations and the needs for improved performance (i.e., document routing and tracking for the Doc team and flight landing control for the Landing team). It appeared that having a specific client, who could continuously provide information, input and feedback, positioned a project high in the continuum of authenticity. The high degree of authenticity motivated the members of the two teams to position themselves as software developers who strived to work to their best to satisfy their clients' needs.

Case 4. The Doc team. In addition to their perceptions that the project was an opportunity to provide solutions for their client, an additional perception was identified among the Doc team members, that is, the team also regarded the OSSD project as a beneficial learning experience preparing them for the future. This perception was shared by all the Doc team members. As Gajra, the Doc team leader pointed out, "The whole point of the class is to get a practical experience of how it is done in the real world." However, this perception was not found with the Landing team members, who were more identified with the role of software developers working to satisfy clients. Perceiving the project as a learning opportunity had stimulated the Doc team members to put in efforts in the documentation task, a task they did not like to work on but was essential for the skill development in software engineering. As described in more details in the next section of this article, the Doc team made special arrangements to ensure that every member of the team obtain maximal learning experience from the project. The team's shared view that the project was a learning opportunity might be partially attributed to the fact that several members were looking for jobs or would be looking for jobs in the near future. Thus, while working on a software product to satisfy a client's needs was an incentive, the Doc team members were also pursuing the project as an opportunity to develop important skills for their future jobs.

With the four cases above, we have illustrated four kinds of perceptions towards the OSSD learning environment, resulting in the formation of four different identities within the context of this project. Figure 1 illustrates the four different identities observed among the students, which were placed in two dimensions in the quadrants chart: project authenticity on one dimension and learning orientation on the other. Although project

Figure 1. Four quadrants showing different identities affected by project authenticity and learning orientation & future goals.



authenticity contributed to the students' perceptions towards the OSSD environment, it was not a single determining factor. For example, for a project of lower authenticity such as Scheduler, Tina saw it as a course assignment that could earn her a grade, while Mark took it as a chance to hone his technical and management skills. For a project of higher authenticity, the Landing team positioned themselves as software developers who were driven by client needs and project goals, while the Doc team not only positioned themselves as software developers, but also took an additional role of being learners. The remainder of the finding section illustrates how the different identities manifested themselves as the students interacted with various components of the OSSD environment.

Teams' Culture

Our initial design of the OSSD learning environment was for each team to serve as users for the other teams, thus developing a learning community across the teams. However, it was found that the cross-team interactions were minimal in terms of user testing and providing feedback to each other. None of the teams left written feedback on the other teams' project websites on GForge. It was observed that for some students the motive in visiting the other teams' project websites was to "make sure we are as good as they are." For some other students, the purpose of interacting with the other teams was mainly to obtain relevant information that could benefit their own project rather than providing useful suggestions to the other teams. Therefore, within this OSSD environment the collaboration appeared to have occurred in one direction, that is, only taking the "good" from the other rather than giving the "good" to the other.

Some students indicated in the interviews that because the members of each team were only concerned and busy with their own project, they did not have sufficient time or they were unable to give sufficient attention to studying and evaluating the other teams' projects. Hence, they were not able to provide useful feedback to the other teams. Lack of time could be a major factor for the lack of cross-team interactions. This piece of data supported our inference discussed earlier that the implementation of communities of practice was confined by physical context and institutional constraints.

Since the main effort for developing software took place within the teams, we now zoom in to examine the learning community within the teams, particularly the interplays between the students' perceptions, their identity, and the community culture exhibited by each team in approaching task delegation and division. The four teams took different approaches towards task division. The Doc team closely followed a specific software development principle—the pair programming principle—based on the Extreme Programming Model (Beck & Andres, 2004), which was suggested by the professor. Two members were paired up to work on a task simultaneously but deliberately alternated roles during the semester. This approach allowed every team member to experience the

software engineering process from various perspectives by playing different roles (e.g., programming, documentation, usability testing, project manager, etc.). It also ensured that the tasks were fairly distributed among the team members. When a member took on a role that required skills beyond his capability, other members took time to teach this member. For example, Neil, a Doc team member, had little knowledge of PHP. Another team member Gary, who was more experienced with PHP, offered to spend time in the computer lab giving tutorials to Neil. The effectiveness of task delegation and the active individual involvement with the learning community experienced by the Doc team may be attributed to the team's shared vision of the project and shared identity as both software developers and learners. The data suggested that Gajra, the leader of the Doc team, played an important role in promoting such consensus within the team. In the interview, Gajra clearly indicated that the project should be a learning opportunity for all, and so she deliberately promoted role rotation in her team:

> . . . if I was always the person responsible for the release, or if I was the person always responsible for doing presentations . . . my role is pretty much defined that I'm learning to do just one thing . . . [It is important to know] what are the different stages of software development, what are the roles of the developer, what are the roles of the project manager, what are the things you have to be mindful of. And it is very difficult to understand it if you are not playing that part. Like if you were not the release coordinator, you wouldn't know how the releases are done. Or if you were not the person responsible for gathering reviews or gathering comments about the documentation and integrating those into your refined document, you wouldn't understand what it takes to do that. . . . by switching the roles we understand each other's responsibility on how it's done and how it's difficult to do each other's role . . . I think it is a better way to do that rather than play one part all the time.

With shared community goals and values, the Doc team members were able to work in orchestra on the software project while maximizing and developing their personal learning experience. In contrast, the other three teams took a different approach in their task delegation and division, which was an approach Gajra had tried to avoid, as indicated in the quote above.

In the other three teams, the leaders typically took upon themselves large portions of the project tasks, such as coordinating various aspects of the projects, whereas the other team members worked on the tasks assigned to them based on their strengths in certain skill areas. For example, if a member was good at a programming language chosen for the team project, he or she was assigned to work on programming from the beginning to the end. From both the observation and interview data, we found no deliberate role rotations taking place in these teams. Most students stayed with the same

tasks throughout the project. If a member could not find a fit in the team project, he or she might feel lost, not knowing how to contribute. In both the Scheduler and Museum teams, there were cases in which individual members could not find suitable tasks that fit their skills until the professor discovered the situation and helped them to find some tasks they were able to work with.

We attributed the differences between the Doc team and the rest of the teams to the presence or absence of a shared perception of their OSSD project among the team members. In the cases of the other three teams, even though some team members took the project as a learning opportunity, not everyone could share the same vision, which resulted in a lack of team-level effort to allow everyone to fully experience different aspects of software development. The lack of shared understanding also led to the uneven distribution of the tasks among the members. We postulated that the shared perception among the Doc team members could be partially attributed to its founder's leadership role, but the founder was not the single determining factor. It required all the members to be able to negotiate meanings and reach consensus in their pursuit of the software project development. The members of the other teams did not reach the same level of shared perception and consensus that the Doc team members did.

Coaching and Feedback

The data indicated that the role played by the professor was very important. In the OSSD learning environment, he acted as mentor for the student teams throughout the semester. Other than giving lectures, the professor also spent a considerable amount of time monitoring the progress of team project by visiting the GForge site to examine the students' projects and their weekly reports. He also met frequently with each of the teams, listening to their reports on the project status, addressing their concerns, and helping some individuals to find a role in the team. For example, the professor spent more time on the two teams that did not involve real clients (i.e., the Scheduler and the Museum) than on the other teams. In addition, the professor also directed the team members to think creatively about their projects, helped them to see what they might have missed in their project, and challenged them to think about new problems and solutions. When the teams got stuck in their projects, the professor would give suggestions or direct the team members to some useful resources. Overall, as we observed, one of the professor's responsibilities was to make sure each team was on task, the workload was reasonably shared, and the project was progressing in a positive direction.

However, the students had various perceptions about the role of the professor. Some students regarded him as a mentor, and therefore appreciated his constructive feedback and encouragement. As Gajra recalled in the interview, when her team's software product was first released, the interface was not very well organized. The professor pointed out the weaknesses in the interface and suggested some resources to help the team improve

the design of the web interface. Following the professor's suggestion, the team created a more user-friendly interface in their second release. The team indicated that the professor's feedback was valuable to their team and the team project.

Some other students, however, had a very different perception of the professor's role. Tina (the Scheduler team) believed that the professor was like a boss, whose main role was to impose requirements on students, such as "you need to get this done by this and this date." Sandy (the Museum team) said, "We feel like we are being monitored [by the professor]" because all the code and documentation of every team's project were stored and openly accessible in the online collaboration system. Under such an impression, she was reluctant to post her team's progress update and project files to the system for the professor to review. Interestingly, Sandy also expressed the concern that there was no "progress track" in this course, meaning no assignments or exams. She said, "you could do nothing all semester, turn in a great project, and get an A . . . (or) you could do stuff all the semester but not finish the project, and not get an A." Apparently, Sandy did not consider the professor's constant monitoring as some form of progress check and his feedback as some form of assessment.

Interestingly, we found that the students' perceptions of the role played by the professor were related to the students' perceptions of the project and their self-identity in the OSSD learning environment. If students considered themselves as learners preparing for a future career or as software developers providing solutions to address clients' needs, then more than likely they would recognize the professor's role as a mentor and appreciate his feedback and guidance. On the other hand, if they viewed themselves as students working for a grade, then they would not want to receive constant feedback from the professor because he would ask them to put in more effort to refine the project. This finding also revealed a conflict between the traditional classroom culture, rooted in the conventional, paper-and-pencil type of assessment system, and the new OSSD learning environment that afforded formative assessment.

Technological System

One of the main features of the OSSD learning environment was the use of an open source software development collaboration system—GForge. This system provided technological affordances that could facilitate team collaboration through useful features, such as a source code repository to store and manage versions of their source code, task management tools to list and prioritize tasks, and forums for project discussion, bug reports, and new feature requests.

We had expected that students would use this system extensively as they engaged in the software development process. However, it was found that the students only used a small portion of the system, mainly the source code repository. Most of the students felt that the repository was very useful because they could save all the source codes and

documents in one place without worrying about the possible file loss. In addition, some students also valued the opportunity to learn and use the GForge system, which was commonly used in the software industry. The level of usage of the other tools varied from team to team. As shown in the GForge system, the Doc team used the system more often than the other teams. Therefore, during the interviews most of the Doc team members were able to describe the specific functions and purposes of the tools in the GForge system. They indicated that the tools were useful to their project, and one student stated that it would be "very inconvenient" if there were no such a system.

However, some students from the other teams, like Sandy, said that the system was "pointless" and they felt obliged to use the tool because the professor required them to use it. Other students, such as Pablo from the Landing team, explained that he was not familiar with some particular technology when asked why he had minimal use of the system. It was true that he was not familiar with the tool, but apparently he was not motivated to learn how to use the system either.

The variation in the students' perceptions towards the technology use could be influenced by their perceptions of the project and their identity. The students who considered themselves as learners or future software developers were more likely to spend time exploring the system and taking advantage of it. The more they used the system, the more they would perceive its usefulness and relevance. By comparison, for those who positioned themselves only as students, whose mere purpose of performing the school work was to obtain a grade, they might feel that the professor had imposed an additional assignment for them to work with; therefore, they were resistant to learning a new tool for their team project.

Discussion

In this study, we took an ethnography research approach to explore students' perceptions and learning experience in an OSSD learning environment. Of particular interest were the OSSD culture and the interplays between the new learning environment and the students' self-processes (e.g., perceptions, experiences, goals, and prior knowledge). The design and implementation of the OSSD learning environment was based on the findings from our previous study on the characteristics of a real OSSD professional community (Ge et al., 2006). While we were trying to implement an authentic OSSD community in this study, the fact that it was implemented in a higher education class setting inevitably placed the OSSD learning environment somewhere between a practice field and a community of practice (Barab & Duffy, 2000). It was a practice field because it was implemented in a school context, in which one of the major goals was to prepare students for the future. It shared many characteristics of project-based learning and PBL. It was a community of practice because it featured real-world projects, real clients, and users, who could interact

constantly with the student software developers with user request, feedback, and suggestions. Due to the uniqueness of the OSSD learning environment, this study has contributed to the research on implementing a community of practice in an educational setting.

The OSSD Environment

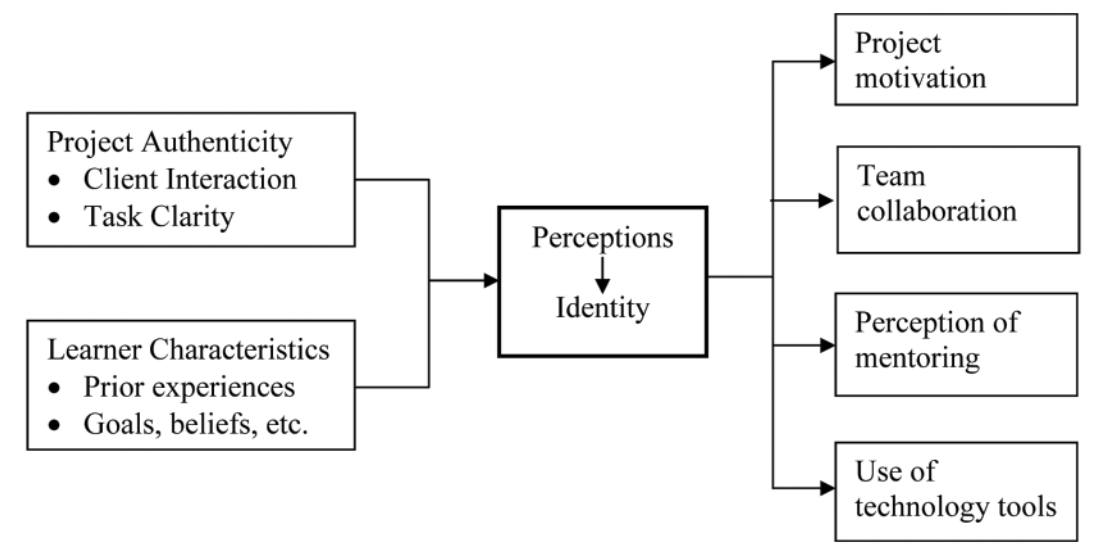
Underlying the notion of community of practice is the assumption that authentic learning tasks, collaboration, and cognitive apprenticeship would facilitate learners' participation and enculturation in the community of practice, which would ultimately lead to the development of usable and robust knowledge (Barab & Duffy, 2000; Brown et al., 1989; Lave, 1988). The findings of this study have led us to critically re-examine this assumption. The findings confirmed the benefits of an OSSD learning environment, which afforded the students with opportunities to collaborate on real-world projects, interact with real clients, and use technology tools to develop software solutions for clients or general public, all of which represented the practice of a software engineering community. The affordances of the OSSD learning environment helped to promote students' identity as software developers as the students engaged in the meaningful service learning process. In addition, the OSSD affordances had also helped to motivate the students to develop their software engineering skills *in situ*, which would enable them to become creative problem solvers, reflective thinkers, and strategic leaders in their future careers.

On the other hand, our study of the OSSD culture has also revealed some complexities and challenges in the OSSD learning environment. This study demonstrated that there was a gap between the ideals and the reality in the process of implementing the OSSD model. The discrepancy could be attributed to various contextual factors and the interplays of those factors in the learning environment. For instance, it was assumed that immersing students in an environment promoting a community of practice could foster intentional learning among learners (Collins, Brown, & Holum, 1991). However, our study showed that this would not necessarily be the case for all the students. We found a number of issues in implementing the OSSD model in the classroom context, for example, the restriction of semester structure, the organization curriculum and courses, the reality faced by students in concerning about getting grades, and the needs for learners to obtain all aspects of learning experiences through rotating roles and tasks instead of dividing up tasks based on individuals' strengths as usually found in a community of practice.

Interplays of Student Self-Processes and the OSSD Environment

Figure 2 illustrates the interactions of those factors or constraints complicating the students' OSSD learning experience, including the authenticity of projects, the nature of tasks, students' perceptions, and the collaboration process and approaches, which all intertwined with one another.

Figure 2. The interplays between project authenticity and learner characteristics led to students' different perceptions of the OSSD learning environment and the development of different identities within the environment, which in turn affected their motivation to work with real-world projects, team collaboration, perceptions of instructor's role, and use of collaborative tools.



The first factor illustrated in figure 2 is project authenticity. A project is often described as authentic or non-authentic. However, our findings suggest that there is a continuum in project authenticity, ranging from higher to lower levels depending on whether or not the project has real, specific clients and users. In our study, the Doc team and the Landing team had specified clients and users. By comparison, the Museum team had a real client, but they had no defined users, and the Scheduler team did not have a real client nor defined users. Project authenticity levels indicated whether the teams had real clients or users to interact with.

Additionally, it was interesting to find that the level of project authenticity seemed to determine the clarity of task definition. We found that for the teams with real clients, the requested software features and functionalities had already been defined by their clients, therefore these students could focus more on developing solutions instead of trying to define the projects. Comparatively, for those teams with less specified users, their projects were less clearly defined and more ill structured. The students had to establish a context and define a problem based on their perceived needs and personal experience. These two types of learning contexts fit Hannafin, Land and Oliver's (1999) classification of "externally imposed context" and "individually imposed contexts" (p. 124). Due to the students' lack of competence in software engineering, we found that the projects with real clients actually scaffolded the students' problem-solving processes.

Project authenticity alone does not determine students' perception of a learning environment. As depicted in figure 2, another determining factor is learner characteristics. As shown in this study, different students responded very differently to the same project. For example, in working with a project with lower authenticity, some students (e.g., Tina) regarded it as merely a task to complete in order to earn a grade, while others (e.g., Mark) took it as a great learning opportunity to experiment with new technologies and practice management skills for a future career. For projects with higher authenticity, while some students (e.g., the Landing team) positioned themselves as software developers, other students (e.g., the Doc team) also intentionally made this project a learning opportunity, in addition to their identified role of software developers. The dual identities of professionals and learners motivated some students to take on additional roles and tasks painstakingly, which were not necessarily required by the project.

As shown in figure 2, the dynamic interactions between project authenticities and learner characteristics resulted in four types of students' perceptions of the project. Consequently, four different identities were developed among the students: a student working to earn a grade, a learner preparing to be future practitioner, a software developer working to satisfy client needs, and both a learner and a developer. These findings were particularly intriguing. When comparing a community of practice and traditional school learning environment, we usually theorize a learner's identity in a duality manner—a student in school versus a contributing member of community (Barab & Duffy, 2000). Thus, in our implementation of a community of practice in school context, we were expecting a transition in identity from being a student toward being a contributing community member. However, our study showed that identity change was more sophisticated than we had expected. We cannot expect all students to acquire the same professional identity by working on some authentic projects. Different perceptions and identities may co-exist due to different learner characteristics. Moreover, being a contributing community member may not be sufficient for achieving the learning purpose. Taking on an additional role of being a learner may help students to benefit more from the new learning environment.

Figure 2 further suggests that different identities, resulting from different perceptions, were manifested as the students interacted with the OSSD environment, leading to different kinds of motivation toward the projects, different collaboration patterns, different perceptions of the professor's mentoring role, and different efforts in utilizing technological tools.

Derived from our findings, figure 2 demonstrates a theoretical model explaining the interplays between project authenticity and learner characteristics, which led to individual learners' different perceptions of the OSSD learning environment and development of different identities within the environment. The students' perceptions and identities subsequently affected their motivation to engage with real-world projects, team collaboration,

perceptions of instructor's role, and use of collaborative tools. We consider this theoretical model as our contribution to the research on implementing community of practice in the classroom setting. We also believe that this study has provided us with insight into the "practice-fields" (Barab, et al., 2002) type of learning environments, including problem-based learning and project-based learning.

Limitation

This study was an exploratory study carried out using an ethnographic research approach. We acknowledged the limitation of generalizing the findings of this study beyond the context of this course and this group of participants. On the other hand, we argue that the research goal for qualitative researchers is to produce a coherent and illuminating description of culture and perspectives on some situations or phenomena based on detailed descriptions rather than to produce duplicated study and results as often found in a quantitative study (Schofield, 2002). Therefore, the main purpose of this study to was to understand the OSSD culture and the participants in a specific context and institution, particularly the individuals' perceptions, experiences, motivation, and behaviors as they made transitions from the traditional culture to the OSSD culture over an extended period of time. However, we believe that similar studies with more participants in other instructional contexts and organizations would help to further corroborate and refine our knowledge on community of practice and to extend the scope of our inquiry.

Implications

This study has yielded numerous practical implications for designing a learning environment approximating a community of practice. The study suggests that when designing a project, authenticity is an important issue to address in such a learning environment. Projects with high fidelity may draw students more closely to reality and promote their identity as a member of a community of practice, which in turn will stimulate and sustain their motivation. When high authenticity projects are not available, the learning environment should be designed to amplify the value and future utility as perceived by students in order to increase their motivation. In addition, scaffolding is essential to help students define problems and clarify tasks when the scope and specifications of a project are not clear. Moreover, explicit instructional support should be provided to help students to understand and make a successful transition to the new learning environment.

This study manifests that implementing a community of practice in a classroom setting can be a complex enterprise. Therefore, learner characteristics must be taken into account when making effort to motivate and scaffold learners to participate in the learning community as contributing members. The intricate culture and complex social system of the OSSD learning environment must be examined in light of the interplays

of various contextual factors and learner characteristics when designing such an open-ended learning environment.

The findings of this study indicate that an open-ended learning environment approximating the community of practice (e.g., OSSD) does not automatically address motivation issues. Thus, efforts must be made to communicate the expectations of the OSSD learning environments to students, help them understand those expectations, and strategically manage and align their perceptions with the goals of a project-based learning environment.

Lastly, because teams are essential units in the cultural system of the OSSD environment, it is important to facilitate team members to develop shared perceptions, understandings, and visions about real-world projects and to align their perceptions and goals with those of teachers or researchers in order to maximize situated learning experience and community of practice.

Future research must focus on how projects with different authenticity levels influence students' task definition, goal planning, and problem solving as they approach projects or problems in a situated learning environment. In addition, more studies are needed to further explore the relationships between project authenticity and learner characteristics in a project-based learning environment, and to examine how changes of a variable in this relationship can have an impact on learners' perception, identity development, motivation, goal orientation, team collaboration, and other variables.

References

Azer, S. A. (2009). Problem-based learning in the fifth, sixth, and seventh grades: Assessment of students' perceptions. *Teaching and Teacher Education, 25*(8), 1033-1042.

Barab, S. A., Barnett, M., & Squire, K. (2002). Developing an empirical account of a community of practice: Characterizing the essential tensions. *The Journal of the Learning Sciences, 11*(4), 489-542.

Barab, S. A., & Duffy, T. M. (2000). From practice fields to communities of practice. In D. Jonassen & S. Land (Eds.), *Theoretical foundations of learning environments* (pp. 25-56). Mahwah, NJ: Lawrence Erlbaum Associates.

Barrows, H. S., & Tamblyn, R. M. (1980). *Problem-based learning: An approach to medical education*. New York: Springer.

Beck, K., & Andres, C. (2004). *Extreme programming explained: Embrace change.* (2nd ed.). Upper Saddle River: Addison-Wesley.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist, 26*(3&4), 369-398.

Boaier, J. (2002). Learning from teaching: Exploring the relationship between reform curriculum and equity. *Journal for Research in Mathematics Education, 33*(4), 239-258.

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

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*(1), 32-42.

Carrington, D., & Kim, S. K. (2003, November). *Teaching software design with open source software*. Paper presented at the 33rd ASEE/LEEE Frontiers in Education Conference, Boulder, CO.

Cognition and Technology Group al Vanderbilt. (1992). The jasper series as an example of anchored instruction: Theory, program description, and assessment data. *Educational Psychologist, 27*(3), 291-315.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator, 6*(11), 38-46.

Creswell, J. (1998). *Qualitative Inquiry and Research Design: Choosing Among Five Traditions*. London, New Delhi: Thousand Oaks/Sage Publications.

Denton, B. G., Adams, C. C., Blatt, P. J., & Lorish, C. D. (2000). Does the introduction of problem-based learning change graduate performance outcomes in a professional curriculum? *Journal on Excellence in College Teaching, 11*(2&3), 147-162.

Eccles, J. S., Adler, T. F., Futterman, R., Goff, S. B., Kaczala, C. M., Meece, J. L., et al. (1983). Expectancies, values and academic behaviors. In J. T. Spence (Ed.), *Achievement and achievement motives* (pp. 75-146). San Francisco: W. H. Freeman.

Feltovich, P. J., Spiro, R. J., Coulson, R. L., & Feltovich, J. (1996). Collaboration within and among minds: Mastering complexity, individually and in groups. In T. Koschman (Ed.), *Computer systems for collaborative learning* (pp. 25-44). Hillsdale, NJ: Lawrence Erlbaum.

Ge, X., Dong, Y., & Huang, K. (2006). Shared knowledge construction process in an open-source software development community: An investigation of the Gallery community. In S. A. Barab, K. E. Hay, N. B. Songer & D. T. Hickey (Eds.), *Proceedings of the 2006 International Conference of the Learning Sciences* (pp. 189-195). Bloomington, IN: The International Society of the Learning Sciences, Routledge.

Ge, X., Thomas, M. K., & Greene, B. A. (2006). Technology-rich ethnography for examining the transition to authentic problem-solving in a high school computer programming class. *Journal of Educational Computing Research 34*(4), 319-352

Geertz, C. (1976). From the native's point of view: On the nature of anthropological understanding. In K. Basso & H. A. Selby (Eds.) *Meaning in anthropology* (pp. 221-237). Albuquerque, NM: University of New Mexico Press.

Geertz, C. (1983). Thick description: Toward an interpretive theory of culture. In R. M. Emerson (Ed.) *Contemporary field research: A collection of readings* (pp. 37-59). Prospect Heights, IL: Waveland Press.

Hannafin, M. J., Land, S., & Oliver, K. (1999). Open learning environments: Foundations, methods, and models. In C. Reigeluth (Ed.), *Instructional-design theories and models* (vol. 2, pp. 115-140). Mahwah, NJ:: Erlbaum.

Hars, A., & Ou, S. (2001, January). *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.

Hmelo-Silver, C. E. (2004). Problem-based learning: what and how do students learn? *Educational Psychology Review, 16*(3), 235-266.

Jeong, H., & Hmelo-Silver, C. (2010). Productive use of learning resources in an online problem-based learning environment. *Computers in Human Behavior, 26*(1), 84-99.

Krajcik, J. S., & Blumenfeld, P. C. (2006). Project-based learning. In R. K. Sawyer (Ed.), *The Cambridge handbook of: The learning sciences* (pp. 317-333). New York, NY: Cambridge University Press.

Lave, J. (1988). *Cognition in practice: Mind, mathematics and culture in everyday life*. Cambridge: Cambridge University Press.

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge: Cambridge University Press.

Liu, C. (2005, May). *Enriching software engineering courses with service-learning projects and the open-source approach.* Paper presented at the 27th international conference on Software engineering, St. Louis, MO.

Liu, M., & Rutledge, K. (1997). The effect of a "learner as multimedia designer" environment on at-risk high school students' motivation and learning of design knowledge. *Journal of Educational Computing Research, 16*(2), 145-177.

Maehr, M. L. (1984). Meaning and motivation: Toward a theory of personal investment. In C. Ames & R. Ames (Eds.), *Research on motivation in education: Student motivation* (vol. 1, pp. 115-144). New York: Academic Press.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis* (2nd ed.). Thousand Oaks, CA: Sage.

Raymond, E. R. (1998). The cathedral and the bazaar. *First Monday, 3*. Retrieved from http://firstmonday.org/issues/issue3_3/raymond/#d1

Resnick, L. (1987). Learning in school and out. *Educational Researcher, 16*(9), 13-20.

Savery, J. R. (2006). Overview of problem-based learning: Definitions and distinctions. *Interdisciplinary Journal of Problem-based Learning, 1*(1), 9-20.

Schofield, J. W. (2002). Increasing the generalizability of qualitative research. In A. M. Huberman & M. B. Miles (Eds.), *The qualitative researcher's companion* (pp. 171-204). Thousand Oaks, CA: Sage.

Spradley, J. P. (1980). *Participant observation*. New York: Holt, Rinehart and Winston.

Vernon, D. T. A., & Blake, R. L. (1993). Does problem-based learning work? A meta-analysis of evaluation research. *Academic Medicine, 68*(7), 550-563.

Wigfield, A., & Eccles, J. S. (2000). Expectancy-value theory of achievement motivation. *Contemporary educational psychology, 25*(1), 68-81.

Xun Ge is an associate professor in Instructional Psychology and Technology at the Department of Educational Psychology, University of Oklahoma.

Kun Huang is a doctoral candidate in Instructional Psychology and Technology at the Department of Educational Psychology, University of Oklahoma.

Yifei Dong is with Openseed Institute, a software company in Norman, OK. He was an assistant professor in Computer Science at the University of Oklahoma when this study was conducted.