Shared Knowledge Construction Process in an Open-Source Software Development Community: An Investigation of the *Gallery* Community

Xun Ge, Yifei Dong, and Kun Huang, University of Oklahoma, 820 Van Vleet Oval, Norman, OK 73019
Email: xge@ou.edu; dong@ou.edu; kunhuang@ou.edu

Abstract: An investigation was conducted to study shared knowledge construction process in an Open-Source Software Development (OSSD) community. Using a qualitative study approach, we examined the resources, tools, and activities in the *Gallery* community to create a rich description of the interrelationships among people, activities, and media. Of particular interest was how the individuals contributed to the building of a shared knowledge base through collaborative problem-solving and decision-making processes, mediated with the OSSD environment. The study indicated that the collaboration process was symbolized by multiple rounds of discussions, which were means to pool individuals' expertise and experiences to obtain quality problem-solving and decision-making outcomes. The study also revealed that the OSSD environment mediated the collaborative efforts through virtual collaboration space, visual organization, and communication tools. The findings have important implications for designing effective instruction specifically for computing disciplines in promoting students' collaborative problem solving and decision making.

Introduction

Our society is in increasing demand of high-quality computing professionals in order to compete successfully in the world market (McGettrick et al., 2004). Computing disciplines—including computer engineering, computer science, software engineering, information systems, and information technology (ACM-AIS-IEEE, 2005) – are highly complex and dynamic disciplines that require students to integrate knowledge from different domains, make connections of various concepts and principles, and develop complex mental models to solve real-world, complex problems. The complex nature of the problems in computing sciences make problem solving difficult and place a heavy cognitive load on learners (Jonassen, 2000a). Thus, computing science education presents a great challenge to educators in this discipline, particularly as the society expects prospective graduates to take on challenges, solve complex problems, generate original ideas, and become life-long learners by constantly self-regulating their learning process and updating new knowledge and skills (McGettrick et al.).

Aside from the complex and ill-defined nature of the domain and the discipline, it is worth examining student learning experience in computing disciplines. It is not uncommon that students perceive taking courses as going through the graduate program and checking off the course list while failing to see the relevance and real-world implications in their future profession. As a result, the knowledge students acquire is often "inert" or "decontextualized" (Bransford, Brown, & Cocking, 2000; Gick, 1986); students often fail to develop a valid and robust knowledge base that will help them to generate "workable" knowledge that can be applied to various situations. This problem raises a serious concern to computing science education, therefore alternative instructional approaches must be sought in order to help students develop the abilities to reason with and apply knowledge to solve real-world problems (Chi & Glaser, 1985; Koschmann, Kelson, Feltovich, & Barrows, 1996), and encourage them to reflect upon their learning process and performance.

In recent years, some educators have directed their attention to the research on open-source software development (OSSD) communities and hope to draw educational implications from the successful OSSD process and experiences. Open-source software is characterized by the free access to the source code that is open to use, modify, and redistribute (Hars & Ou, 2001). In developing open-source software, a community of dedicated professional developers and user participants volunteer their expertise and energy over a long period of time toward a common goal. By utilizing online work space and communication tools, these geographically distributed participants collaborate through continuous meaning negotiation and knowledge sharing. Raymond (1998), in his work "The Cathedral and the Bazaar," has described OSSD community as "a great babbling bazaar of differing agendas and approaches … out of which a coherent and stable system could seemingly emerge only by a succession of miracles" (¶ 4); yet it is this kind of "bazaar" that produced successful software like Apache, Linux, Mozilla Firefox.

An OSSD community is a typical example of communities illustrated by Lave and Wenger (1991). It is a well-defined identifiable group involving a collection of individuals actively and mutually engaging in an activity system, sharing repertoire in the pursuit of a shared enterprise over an extended time frame (Barab & Duffy, 2000; Wenger, 1998). Past literature on community of practice and learners have offered valuable implications for designing meaningful learning environments (e.g., Barab & Duffy, 2000; Wenger, 1998, Lave & Wenger, 1991), and the research on activity theories (e.g., Engeström, 1987; Barab, Evans, & Baek, 2003; Jonassen, 2000b) has provided us with another lens to characterize and analyze the community as a participatory unit of activity closely (Barab et al.). However, it is believed that as an activity system, every community has its own characteristics, and so does an OSSD community. Hence, it is necessary to examine the common characteristics of OSSD communities in light of theoretical frameworks such as communities of practice and activity systems so that we can intentionally design and create a learning environment with those specific characteristics for target learners. Of particular interest are questions such as: What are the unique features of the OSSD environment that lead to the success of open-source software? What motivates the participants to persistently pursue their goals? How do the participants collaborate in the process of decision making and problem solving?

While a few studies have been conducted to explore various aspects of OSSD by researchers from various disciplines and for different purposes, including motivation (Hars & Ou, 2001), trust and control (Gallivan, 2001), community joining (Krogh, Spaeth, & Lakhani, 2003), we have not encountered studies examining OSSD communities in terms of problem-solving and decision-making patterns, the cognitive distribution among the community developers, and the building of a shared repertoire in the community. Based on Jonassen's (2000) work, we would like to further examine how the authenticity and the complexity of problem solving (e.g., decision making) in the OSSD context maximize the natural collaborative processes of community participants (Nelson, 1999), how distributed cognition in the OSSD community contributes to multiple perspectives, decomposition of problems, and reduction of complexity. In understanding that a community is a sub-system of an activity system, we are interested in investigating the activity system triangle in the OSSD community to account for meaningful participants, processes, tasks, goals, and transactions: how the "subjects," the OSSD members negotiate and mediate rules and customs and use the "tools" and the technology system to work on the "objects," such as software programs.

A thorough understanding of the OSSD community may help us to design an effective simulated OSSD environment for computing courses aiming at facilitating students to integrate knowledge and develop shared understanding while collaboratively engaging in reasoning, problem solving, and decision making in real-world situations. Therefore, our research is aimed at studying the collaborative problem-solving and decision-making processes of an OSSD community in the hope of drawing some valuable implications to inform the teaching practice in computing disciplines. Specifically, we are seeking explanations to the following questions:

1. How are problems represented, solutions negotiated and reached among the OSSD community members?
2. How do the OSSD community members distribute cognition and contribute to the building and development of shared knowledge?
3. What features does the OSSD environment afford to facilitate collaboration process and the software development among the community members?

Method

As the main purpose of the study was to study how the professional developers communicated and collaborated to perform problem-solving tasks and make decisions in the specific context of the OSSD environment, a case study approach was employed in order to investigate the complex social phenomena holistically and meaningfully within the real-life context (Yin, 2003). As there are countless OSSD communities in the public domain, each containing massive amount of data, including numerous mailing lists, messages, archives, and program codes, we had to sample and focus on one community to conduct an in-depth examination. In this case, an OSSD community was treated as a single case with multiple units of analysis (Yin).

Case Sampling

We used a purposeful sample strategy with a set of criteria to identify the case to be studied. Consequently, the OSSD community of "Gallery" was selected from www.sourceforge.net on the following criteria: activity level, scale level, and maturity. The sampling rationale is that by looking for a high activity level, small scale, and relatively mature project, we are able to observe and examine the full-cycle software development processes in an active, quality OSSD community whose scale is comparable with a typical classroom environment. Thus, the

Gallery community met our criteria with its 99.98 percentile all time activity, thirty-four actively working developers, and its current stable production stage.

Data Sources

According to the introduction on its website, "Gallery" is a "slick, intuitive web based photo gallery with authenticated users and privileged albums. It is easy to install, configure, and use." It started to be an open source project from July 2000 after the initiating developers of the project posted the first round of the code. In 2001, dissatisfied with Gallery version 1, some developers began to develop Gallery version 2 (referred to G2 in later sections) from scratch.

On the Gallery website, a variety of tools and resources are available to facilitate the communication and collaboration among the developers and the users, including *IRC*, *CVS*, *Bug report, Feature request, Patches, Tasks,* and *News*. Some of the tools and resources are mainly used by developers to coordinate their efforts of developing software, while others are mainly for users to provide input and feedbacks to the developers about the software. All these resources became our rich data sources. Each of the tools or resources was described below in relation to its functionality and its use by developers and other users:

- *IRC:* a chat channel for weekly real time communication among the developers.
- Six mailing lists for developers and subscribers.
- *CVS* (Concurrent Versions System): a source code tree for the developers to manage code changes.
- *Bug report*.
- *Feature request*: for users to request the features they would like to see from Gallery.
- *Patches*: for advanced users; the fixing code – patches are posted to this list.
- *Tasks*: added by a developer who identifies a bug to fix or a feature to work on and update progress.
- *News*: provides information about the latest Gallery releases.

Data Source and Analysis

For the current study, we decided to use the *–devel* mailing list as our major data source, supported with the other mailing lists and archived sources as mentioned above. The reason is because *-devel* is a major communication tool for Gallery developers to discuss technical issues in Gallery development. In addition, this mailing list has archived all the email communications from November 2001 till the present day and the archive is readily accessible. Every email message in the mailing list contains author's name, date and time, subject, and message content.

We, the three researchers of this study, consisting of two researchers in instructional psychology and technology and a professor in computer science, worked together to observe and code the data. First, we took observation notes of our overall impressions of the Gallery environment by looking into different data sources and areas, and then we focused on the postings in the *–devel* mailing list for detailed and thorough coding. In addition, we frequently referred to the *–checkins* list and *CVS* in order to find the modifications made to source code as a result of some discussions in the *–devel* mailing list; the *bug report* and *feature request* in order to identify what triggered certain discussion at certain point of time; and the *news* archive in order to link release news with certain discussions among the developers. During the entire data analysis, we went through the processes of open coding, axial coding, and selective coding in order to abstract meanings, infer patterns, generate categories, draw hierarchical relationships among the categories, which result in the integration and refining of a larger theoretical scheme with themes to answer the research questions (Strauss & Corbin, 1998). We often discussed our interpretations, negotiated meanings, and finally reached consensus on all the codings. The professor of computer science provided not only explanations on many technical terms, but also unique insights into the understanding of the data in the process of data analysis, which greatly enhanced the validity of the data analysis.

Findings

The preliminary findings based on 42 emails from six developers, which we have examined thus far, provided us with some snapshots of the shared knowledge construction process of the Gallery community, their collaboration, interactions, problem solving, and decision making. The complete data analysis will be included in the final full-length paper and presented at the conference. The following characteristics were descriptive of the themes emerging from the data: (1) careful project planning by core members at the initial stage; (2) multiple rounds of discussions in the processes of problem representation and solution; and (3) the affordance of the OSSD environment in facilitating collaboration among the community members.

Careful and Systematic Project Planning

Systematic project planning and management is clearly evident in the process of the Gallery 2 (G2) development. In the initial stage of G2 development, D1, one of the first co-founders/initiators of Gallery, took the initiative to present a systematic plan for G2. He sent a series of emails to the *-devel* mailing list, presenting a "development roadmap" for the project. The roadmap essentially decomposed the project planning into three categories: feature set, release schedule, and development branches. In addition, D1 not only discussed version G2.0, but also envisioned the development of G2.1, G2.2, and versions beyond. Furthermore, he prioritized the development tasks to be incorporated in different versions. As one of the developers commented, "thank you for setting up a way to discuss Gallery on a more technical layer." By presenting his initial ideas in these three categories, D1 has laid out a concrete starting framework for the G2 project for the other developers.

As an essential feature of the Gallery community, careful and systematic project management served as a blueprint, which set up a common ground for the community members to collaborate in problem representation, solution development, and decision making. The data also indicated that core developers played an important leadership role that was critical in ensuring the smooth workflow and the success of the project.

Multiple Rounds of Discussion for Problem Representation and Solution

As observed, one of the most salient features of the Gallery OSSD community was multiple rounds of discussions, which was characterized by sharing ideas, negotiating meanings, expanding an original thought, clarifying misconceptions, proposing alternative perspectives or solutions, and reaching consensus. Such discussions facilitated problem representation, solution development, and decision making.

Problem Representation

After the planning stage, the Gallery team moved on to the process of defining and representing the problems via multiple discussions. This process consisted of invitation for input, knowledge sharing, meaning negotiation, and consensus reaching, with various outcomes regarding shared knowledge construction process, for example: individuals' ideas were interconnected, expanded and integrated to form new knowledge; knowledge gaps were filled; the nature or the scope of the problem was further defined so that the problem became clearer. The outcomes of the collaborative efforts were illustrated through the following examples.

Although D1 has drawn the initial roadmap for G2 development in his initial post, he was seeking further input from the other developers. In fact, his initial message served as an invitation for further discussion, refinement, and revision of the identified tasks and problems. For instance, after another developer responded to D1's message with some of his thoughts, D1 replied, "Like I said we haven't put much thought into this. It would be really cool if someone would though. And you seem to have a lot of ideas :) :). Hint hint..." ("Gallery," 2005)

The Gallery developers demonstrated more holistic consideration when they were defining a problem. Through discussion, D2 realized that the "workflow" and the "wizard" were actually interrelated. He wrote:

> "I fear that workflow and the new config wizard need to harmonize. Unless I know how the new wizard works I can't specify the workflow. But the wizard probably needs to know more about the workflow before going into wrong directions. How do we elaborate this topic? Let's start with user management at first, because that's the common base?"

Apparently the developer were considering two interrelated factors at the same time, and identified a "common base" of the two in order to facilitate the discussion and the development of both.

Some of the initial ideas presented by D1 were rather vague or did not have a clear focus, but the other developers further elaborated and refined those ideas in their responses. For instance, when D1 proposed the new feature for G2 – creating album hierarchy in batch manner, D3 quickly pointed out that this was in response to the problem that the photo uploading process took too long. He then clarified that "the demand is, to import images in a very fast way, maybe even w/o thumbs," and he suggested finding "an 'offline' way to generate the thumbs and highlights in the background," which would not take up the uploading time.

Developing Solutions

Sometimes when a developer shared his/her initial thought, other developers would agree upon and support it; in addition, they would add further information to expand the accepted idea proposed by the first developer. For example, when discussing G2's workflow management, D1 proposed some of the future workflow features for the software. Other developers responded with more workflow features. D3, for instance, wrote in his response, "This is only the half of it, the goodies." He further explained what other workflow features G2 could afford. As a result of other developers' contribution, a more detailed picture was drawn with regard to G2's workflow management.

When some developers suggested solutions to a problem, others might question their feasibility by reminding them of some potential problems or possible constraints that might affect the realization of the solutions. This was illustrated by the discussion of the image firewall feature, in which D3 thought that PHP, a scripting language, would easily solve the problem. However, D1 replied, "It is in fact pretty easy. But there are some issues. It breaks shutterfly integration. It also breaks our current mirroring capability. We gotta work this stuff out."(Shutterfly is an online photo service provider that G2 was linked to.)

D3 suggested allowing unrestricted access to G2 from Shutterfly, but then he quickly realized that doing this would open the door to hackers. Later on, other developers suggested alternative ways to avoid hackers, with constraints discussed for each of the solutions. By providing different perspectives and sharing their own experiences, the developers in the OSSD community collaborated to avoid potential failures that could have otherwise resulted due to an individual's limited knowledge or experience.

Usually, when a developer raised concerns about the constraints of a solution, alternatives were also suggested. When discussing the interface of the G2 album, D1 first suggested using multi-tabbed interface with "no more popup windows." However, D3 didn't think the multi-tabbed interface was a good idea because it would interfere with the "What You See Is What You Get (WYSIWYG)" effect of the interface. He suggested using a single popup window and explained how the popup window could be connected to the gallery main interface. D1 replied back asking D3 to elaborate his reasons and inviting him to view his actual multi-tab design, which he believed would work very well. Meanwhile, D1 also reminded D3 that one of the goals of G2 was to get rid of all the JavaScript dependencies (popup window is JavaScript dependent). At the conclusion of this round of discussion, D3 dropped his idea and acknowledged that popup windows might not work in that particular environment.

The OSSD Environment Affordances

Since Gallery users and developers are geographically scattered, they have to rely solely on a virtual space afforded by technology – the website and its associated database – to communicate and collaborate. Therefore, technology plays an essential role in this virtual community space for the software product development to be successful, as evidenced in the findings that have been presented above.

The visual representation of the data and the resources in categorized and hierarchical order enable the Gallery community members (both developers and users) to organize information, augment thinking, and represent problems for effective collaboration. This is illustrated by various mailing lists, each of which is visually categorized, such as *–devel* and *–checkins,* serving a different purpose. Within the *–devel* mailing list, discussions are broken down into individual topics so that it is easy to track emails regarding a specific topic. Another example is the *CVS* repository, which provides a holistic visualization of the Gallery source code by showing the hierarchical structure of the code, thus helps to coordinate the team efforts.

The collaboration space and the communication tools in the Gallery website allow the members to communicate effectively with one another via chat or emails, upload or download documents, submit and examine source codes, and elicit input and provide feedback. The developers may make modifications to a source code which will then be uploaded to *CVS*. As soon as the source code is updated in CVS, the developers will get instant email notification of the new changes. Further, the collaboration space and tools enable developers to gain continuous information and direct feedback from the Gallery users by browsing the users' *bug reports*, *feature request*, and *patch* list and discussing and helping users in the Gallery forums. The input and the feedback from users help the developers to make decisions on what new features proposed by users must be included and worked on. The communication tools also help the developers to continuously update their work progress to one another and to users through the "*tasks*" list.

The starter kit and the archives in the Gallery website also serve as historical heritage of the Gallery community. The starter kit is intended to acquaint new developers with the Gallery development team, the current operations, procedures, coding styles, and design philosophy. Meanwhile, the archives allow the current developers to track the historical development of each project and encourage new members to track back for any questions they may have for the project.

Discussion and Implications

From the initial problem presentation to the final solution of the problem, individuals in the Gallery community contribute to the building and shaping of the community knowledge. Not a single member in this community possesses all the expertise, but with different background, experience and expertise, each member brings to the Gallery community his/her unique perspective to advance the development of the Gallery software. As essential collaborative activities, the members of the Gallery community (both developers and users) bring multiple factors into consideration supporting, questioning, or challenging an idea. Consequently, through collaboration, distributed cognition in the Gallery OSSD environment becomes cumulative wisdom, which every individual has access to. The whole is greater than the sum of parts, as argued by Bell and Winn (2000).

As suggested by this study, all the collaborative efforts in the Gallery community would be impossible without the mediation of the technology in the OSSD environment. The OSSD technology system plays an inseparable role in extending human capabilities (in both individuals and the community), distributing cognition, coordinating collaborative efforts, and building shared knowledge. As Pea (1993) observed, "tools literally carry intelligence *in* them" (p. 53).

Although this study is not conclusive yet, the current findings do provide some useful design implications for integrating the OSSD approach into computing education. First, the OSSD environment is a powerful cognitive modeling system, in which students not only learn from professional developers how to reason, solve problems, and make decisions concerning real-life projects, but also how to use the resources and tools afforded by the OSSD environment and collaborate with peer developers and users to work toward the common goal of developing quality software. Students can learn, by observing core developers, how to define and represent problems, how to negotiate meanings to generate ideas for problem solutions, and how to conduct user and task analysis, and how to develop a careful plan that will help them to reach every milestone. All the OSSD activity is helpful for students to write a Request for Proposal (RFP). The multiple rounds of discussions in the processes of problem representation and solution by the core developers imply that the instructor should encourage students to engage in numerous team discussions, value different perspectives and facilitate shared knowledge building processes. The instructor should also help students to take advantage of all kinds of tools and technological features of the OSSD system to collaborate closely with team members.

Second, the OSSD environment that emphasizes the processes over the products is a beneficial instructional approach for computing education. Students can either participate in an existing OSSD community or in one set up by the instructor, depending on specific instructional needs and students' prior skill levels. If students are novice software engineers with little real-world programming experience, it is recommended that a small OSSD environment be created for the course. Students work in teams on real projects requested by real clients, assuming the roles of *developers* for their own team and of *users* for other teams while learning how to adaptively apply their knowledge to solve real-world problems. Under the instructor's guidance, the community will grow beyond the course and over time as new members join in the OSSD community in subsequent semesters or years. As in any of the real OSSD communities, newer members will inherit the community's culture, rules, customs, artifacts and so on, and continually work to better and upgrade software programs. Students in a more advanced level may be encouraged to join in a real OSSD community and learn to think and perform like real computer programmers.

Further, the design of an OSSD environment for computing education can be integrated with the idea of partnership with computer industries, companies, and other universities, as suggested by McGettrick, et al. (2004), which will mutually benefit all the parties involved.

We recognize that there may be differences between an OSSD community created for class use and an existing, spontaneous OSSD community in terms of different levels of engagement and different degrees of authenticity. Therefore, we are in the process of investigating the effects of a course-based OSSD community established by an instructor in an instructional context to be compared with an existing OSSD community

developing spontaneously and naturally in a social context. We believe that interesting findings may emerge from these studies.

References

ACM-AIS-IEEE. (2005). Computing Curricula 2005, a cooperative project of the Association for Computing (ACM), the Association for Information Systems (AIS), and the Computer Society (IEEE-CS). Retrieved November 11, 2005 from http://www.acm.org/education/curricula.html.

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. 25-55). Mahwah, NJ: Lawrence Erlbaum Associates.

Barab, S. A., Evans, M. A. and Baek, E.-O. (2003). Activity theory as a lens for characterizing the participatory unit. In D. H. Jonassen (Ed.), *Handbook of Research for Educational Communications and Technology* (2nd ed., pp. 199-214). Mahwah, NJ: Lawrence Erlbaum.

Bell, P., & Winn, W. (2000). Distributed cognitions, by nature and by design. In D. Jonassen & S. Land (Eds.), *Theoretical Foundations of Learning Environments* (pp. 123-145). Mahwah, NJ: Lawrence Erlbaum Associates.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn: Brain, mind, experience, and school*. Washington, D.C.: National Academy Press.

Chi, M. T. H., & Glaser, R. (1985). Problem solving ability. In R. J. Sternberg (Ed.), *Human abilities: An information processing approach* (pp. 227-250). New York: W. H. Freeman and Company.

Engeström, Y. (1987). *Learning by expanding: An activity-theoretical approach to developmental research.* Helsinki: Orienta-Konsultit.

Gallivan, M. J. (2001). Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies. *Information Systems Journal, 11*, 277-304.

Gick, M. L. (1986). Problem solving strategies. . *Educational Psychologist, 21*(1-2), 99-120.

Hars, A., & Ou, S. (2001). *Working for Free? – Motivations of Participating in Open Source Projects.* Paper presented at the The 34th Hawaii International Conference on System Sciences, Hawaii.

Jonassen, D. H. (2000a). Toward a design theory of problem solving. *Educational Technology Research & Development, 48*(4), 63-85.

Jonassen, D. H. (2000b). Revisiting activity theory as a framework for designing student-centered learning environments. In D. Jonassen & S. Land (Eds.), *Theoretical Foundations of Learning Environments* (pp. 89-121). Mahwah, NJ: Lawrence Erlbaum Associates.

Koschmann, T., Kelson, A. C., Feltovich, P. J., & Barrows, H. S. (1996). Computer-supported problem-based learning: A principled approach to the use of computers in collaborative learning. In T. Koschmann (Ed.), *CSCL: Theory and practice of an emerging paradigm* (pp. 83-124). Mahwah, NJ: Lawrence Erlbaum Associates.

Krogh, G. v., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy, 32*, 1217–1241.

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

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

Nelson, L. M. (1999). Collaborative problem solving. In C. M. Reigeluth (Ed.), *Instructional-design theories and models: A new paradigm of instructional theory* (Vol. 2, pp. 241-267). Mahwah, NJ: Erlbaum.

Pea, R. D. (1993). Practices of distributed intelligence and designs for education. In G. Salomon (Ed.), *Distributed cognitions: Psychological and educational considerations* (pp. 47-87). Cambridge, UK: Cambridge University Press.

Raymond, E. R. (1998). The cathedral and the bazaar [Electronic Version]. *First Monday*, 3. Retrieved November 11, 2005 from http://firstmonday.org/issues/issue3_3/raymond/#d1.

Strauss, A., & Corbin, J. (Eds.). (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory.* Thousand Oaks, CA: Sage Publications.

Wenger, E. (1998). *Communities of Practice: Learning, Meaning, and Identity*. Cambridge: MA: Cambridge University Press.

Yin, R. K. (2003). *Case study research: Design and methods*. Thousand Oaks, CA: Sage Publications.