

Dimensions and Forms of Knowledge Collaboration in Software Development

Yunwen Ye^{1,2}

¹*Department of Computer Science
University of Colorado
Boulder, CO80309-0430, USA
yunwen@cs.colorado.edu*

²*SRA Key Technology Laboratory
3-12 Yotsuya, Shinjuku
Tokyo, 160-0004, Japan*

Abstract

Nowadays few software systems can be produced by a single software developer. Most software systems require collaborative development due to the required brawn power brought by multiple hands. At the same time, because software systems are knowledge artifacts that involve knowledge from multiple domains that a single software developer often does not possess, collaboration also becomes a necessity due to the required brainy power brought by multiple minds. Merging the above two lines of requirements, we view knowledge collaboration as an essential elements in future software development technology and support environments. This paper discusses different forms of knowledge collaboration that exist in software development and suitable conditions for successful knowledge collaboration.

1. Introduction

Modern software systems often contain tens of thousands or millions of lines of code, and it has become hardly possible for a single software developer to hack out such complicated and large systems; they have to be developed through the collaboration of many software developers. In other words, collaboration has become the essential part of software development, and distributed software engineering has become an important research fields in software engineering [1].

However, most of the research has focused on the brawny aspects of collaboration, namely, the power brought by many hands. The major concerns have been on the cooperation, communication, and coordination problems brought by the consequences of division of labor. The labor of development software systems might be divided along peoples in a small group of one organization, across different project teams, across different sites, or across different organizations.

Another aspect of collaboration in software development that has not been paid enough attention is

the brainy power brought by multiple heads of software developers, namely the knowledge collaboration that takes place during the process of software development.

Software development is a knowledge-intensive and distributed cognitive activity [2]. It requires knowledge from several different domains, including both the computing domain and the application domain.

As computer applications are getting more complicated and larger, the amount and kinds of knowledge required get more complicated. Even if the software itself might be a simple one, because the computational environment in which it runs consists of large and complicated programs, even writing a simple software system still demands substantially more knowledge than applications written in early days. For example, 20 years ago, a simple C program running in a Unix environment would only interact with the operating system. Some knowledge about hardware, the operating system, C libraries (which are rather small), the C language, data structure, algorithms, and the application domain would suffice.

Nowadays, a typical Java program runs in a more complicated environment. Because a Java system often has to interact with frameworks, Java API libraries, Java Virtual Machine and native libraries, a good Java developer needs to have knowledge about all of the above in addition to all the knowledge required of a good C developer. All the underlying systems are very complicated, and entail a long learning process.

Few developers, if any, have all the knowledge needed in their own heads. The knowledge necessary for software development is distributed between the developer and the external world, and the development of a software system requires learning from and integrating the knowledge of various external sources in the world. Knowledge in the world comes from cognitive tools that support programming and knowledgeable collaborating peers.

The development of software is therefore no longer confined to an individual software developer but has to rely on distributed cognition by reaching into a complex networked world of information and computer mediated collaboration.

With the current trend of globalization of companies, software companies are increasingly become distributed along different times zones, locations, and cultures. The distribution of software companies has become necessary not only because of the needs of shifting labors to places that have lower costs, but also because of the pursuit of local talents that are otherwise unavailable. In other words, in addition to delegate the task of development to the most economically viable places, which is the current driving force of outsourcing, software development companies need also to ship the task to the most talented and suitable people regardless of location, time zone, and national boundary, which will soon be the upcoming driving force of outsourcing. Knowledge-based collaboration has become as important as, if not more than, the current labor-based collaboration in software development.

This paper defines the concept of knowledge collaboration, discusses different forms of knowledge collaboration that exist in software development and analyzes suitable conditions for successful knowledge collaboration, in order to lay a theoretical foundation for the development of tools and environments that support knowledge collaboration in software engineering.

2. The Concept of Knowledge Collaboration

2.1 Knowledge and Knowing Capability

The notion of knowledge has expanded from its traditional focus on knowledge as expertise “stored as artifact” in a person’s head [3] to the focus of a person’s ability to find one’s way around the world, through effective interaction and collaboration with tools and people in his or her environment. To achieve efficiency in knowledge work, including software development and design, knowledge workers need not only utilize the knowledge that they have acquired in the past, but also exert their *knowing capability* in practice—the ability of accessing and learning from external knowledge resources in the context of problem solving. This expansive perspective of knowledge leads away from the attention on what a knowledge worker knows in abstract that have been the major focus of current discourse on software engineering education and knowledge management, and towards a focus on the situated actions taken by people when they engage in work, and the intelligent behaviors demonstrated by them in utilizing whatever internal and external resources available in practice [4].

The idea of knowing capability in practice is grounded in the theoretical thinking of philosopher Polanyi [5], who notes that most of human knowledge is tacit; sociologist Giddens [6], who argues that immense knowledgeability is involved in the conduct of everyday activity of human beings; design theorist Schon [7], who finds that

designers demonstrates and learns by continuously reflecting in their actions; biologists Maturana and Varela [8], who argue that knowing is manifested as effective action in the domain of existence of other living beings. Those theoretical thinking are further supported by anthropological studies of Lave [9], Hutchins [10], Orlikowski [11], and Suchman [12].

The focus shift from individual knowledge to knowing capability in the context of human interaction with tools and peers has led to the development of the theory of distributed cognition, which views an individual and their surrounding context as a whole cognitive system [2] to be analyzed and supported. Hollan et al. points out that cognitive processes are distributed along at least three axes:

- (1) Cognitive processes may be distributed across the members of a social group.
- (2) Cognitive processes may involve coordination between internal and external (material or environmental) structure.
- (3) Cognitive processes may be distributed through time in such a way that the products of earlier events can transform the nature of later events.

In the domain of software development, software developers use computational tools to develop computational systems, and the results of their earlier development events are also embedded in the development tools. Therefore, the distribution is basically along two axes: the social axis (with other knowledgeable peers) and the technological axis (with development tools as cognitive aids).

Knowledge collaboration, a joint intellectual endeavor in which a knowledge worker engages with cognitive tools and peers, has thus become an essential activity in most of today’s software development, and it defines the knowing capability and competence of software developers.

2.2 Two Forms of Knowledge Collaboration

Following Moran and Ghoshal [13] who argues that all new resources, including knowledge, are created through two generic processes: exchange and combination, we view knowledge exchange and collaborative knowledge construction as two basic forms of knowledge collaboration.

Knowledge exchange takes place when knowledge held by one person is transferred to others. Knowledge exchange does not necessitate direct interactions when knowledge can be codified and stored in knowledge repositories, and the knowledge seeker is able to retrieve and applied it in his or her work. The creation and reuse of reusable components, and the identification and use of design patterns are two examples of knowledge exchange-based knowledge collaboration in software development.

However, the other major form of knowledge, tacit knowledge [5], cannot be readily exchanged through knowledge repository. Because tacit knowledge is fundamentally embedded in everyday practice and social interaction, it can only be transferred through social interaction and co-activity. In such cases, direction communication and interaction between knowledge seekers and knowledge providers have to take place for the tacit knowledge become accessible and transferable.

Collaborative knowledge construction occurs when the knowledge workers cannot cover all the spectrum of the knowledge required to solve a problem. New knowledge has to be created by combining knowledge held by each worker. Collaborative knowledge construction is not a simple sum of each participant's knowledge, but a result of mutually stimulating exchange embedded in social interaction and practice. During this process, implicit assumptions are externalized; different viewpoints are presented, negotiated, and integrated; formerly unconnected knowledge elements get connected and combined.

3. Roles of Computers in Supporting Knowledge Collaboration

As mentioned earlier, software development process is distributed long two axes: the technological axis in which computer plays the role of collaborating partner, and the social axis in which developers work together to develop a software system. Accordingly, computers can play two roles to support knowledge collaboration: as a cognitive tool to support software developers and as a mediating tool to provide a platform for knowledge transfer and combination among software developers.

3.1 Collaborating with Computers

Since the early days of computing, computer-based tools such as compilers and reusable libraries have always been a quintessential part of the repertoire that consists of the technical competence of software developers. However, such tools have not become an equal partner in the underlying knowledge collaboration process of software development, and their contribution to software development is passive and indirect. Software developers use such tools by delegating a part of development tasks such as compiling and linking, or by utilizing them to carry out a planned action such as editing. An equal partner in the knowledge collaboration process of software development should directly contribute their own knowledge to the development process, to complement the insufficient knowledge of software developers.

An example of research in computer-aided software development environments that view computers as equal

partners in the knowledge collaboration process of software development is the LispCritic system [14], which is an intelligent agent that helps programmers improve their programming skills. It uses program transformation rules to suggest a syntactical equivalent that is either a more cognitively efficient or computationally efficient solution after it has recognized a less ideal code segment. By using LispCritic, software developers can improve their skills by working together with the system that provides feedback on their sub-optimal solutions.

The idea of developing intelligent agents to assist software development has been carried further in the development of the *Argo* design environment [15]. *Argo* is equipped with computer critics that deliver general software design knowledge for software developers to reflect upon their current design based on the recognition of insufficient knowledge demonstrated by software developers in their current development.

CodeBroker [16] is another system that we have developed to be a programming partner with software developers. CodeBroker uses user modeling techniques to analyzing what software developers know about Java API libraries from their previously constructed Java programs, task modeling techniques to infer what they need to know for the construction of current program by analyzing what they are doing in the development environment, and then actively deliver task-relevant reusable library components that they do not know yet to the development environment. By using CodeBroker, software developers are able to utilize knowledge in external resources (reusable libraries) in a way similar to the knowledge about reusable components in their heads.

3.2 Computer-Mediated Knowledge Collaboration

In computer-mediated knowledge collaboration, computer plays the role of connecting software developers in need of new knowledge with software developers who have the knowledge, instead of directly contributing to the knowledge creation process of software development. Computational environments provides a socio-technical platform for software developers to exchange and combine knowledge from different resources. This line of research has been mainly conducted in CSCW and CSCL.

Because traditionally software development has been viewed as an individual cognitive process in software engineering, applying CSCW and CSCL research results to support software development has been quite rare. The Answer Garden system [17] is an early effort in supporting Unix users to obtain help from other knowledgeable peers by routing questions asked by a user to pre-registered Unix experts. The system is not tightly coupled with software development itself. Terveen et al.

developed a similar living design memory system [18] that connects software developers to share knowledge about software design.

Expert Finder [19] is a system that directs a question posed by a Java developer to another developer that has the matching expertise, so the expert developer can provide help through emails. Expertise is mined from analyzing the programs that Java developers have written. Expertise Browser [20] is another system that mines the CVS logs to find who have expertise on the portion of a developed software system, so that the maintainer can direct his or her question to those experts.

All of the above systems have focused on finding the experts on a particular problem or a domain. An important aspect that has been missing is why the experts should participate in the knowledge collaboration process. For a knowledge collaboration process to succeed, finding who are experts is only the first step, and the second step is those experts must be willing to offer help. To create computational platform to support knowledge collaboration, the motivation to participation in knowledge transfer and combination has to be considered from the beginning. This is the major reason that computer-mediated knowledge collaboration has not been very successful yet, and the remaining of the paper will be dedicated to the understanding of creating successful environments for knowledge collaboration.

4. Dimensions of Knowledge Collaboration

To create computational environments to support knowledge collaboration, we first need to understand what are the prerequisite conditions for successful knowledge collaboration. We argue, in this section, that knowledge collaboration is easier to take place when participants share proximities in three dimensions: cognitive, structural, and relational. Cognitive proximity refers to overlap of the participants' intellectual interests, which lays the common ground for meaningful knowledge exchange and combination. Structural proximity refers to the easiness of mutual accessibility to the knowledge of each participant, which creates the possibility for collaboration. Relational proximity refers to the social closeness and tie that participants feel about each other, which provide motivations for participation and lubricate the process of knowledge collaboration.

4.1 Cognitive proximity

Two people share cognitive proximity about a problem if they both have overlapping knowledge required for the solution of the problem, and attach similar values to the solution of that problem as well as to the new knowledge that might be created during the process of problem solving through collaboration. To notice that the definition of cognitive proximity is contingent on a

problem. Only after a problem around which knowledge collaboration is needed is defined, can the cognitive proximity become defined and measured.

Overlapping knowledge provides a common conceptual apparatus for knowledge collaboration, and determines the knowledge absorptive capacity that refers to the ability of recognizing the value of new knowledge and of interpreting, assimilating and using it [21]. Knowledge collaboration is achieved through communication. Efficient and meaningful communication is enabled by the common ground that communication partners share and are aware that they share. The shared common ground makes communication possible by enabling the mutually intelligibility of their communication acts. The mutual awareness of the shared knowledge makes communication efficient by reducing the amount of information to be communicated.

Complete overlap of knowledge is detrimental to successful knowledge collaboration because it is the asymmetry of knowledge and knowing capability that necessitates the needs of and creates the opportunity for knowledge exchange and combination. Cognitive proximity, therefore, refers to not only the extent of the overlapping knowledge held by collaborating partners but also the extent of the complementariness of the knowledge held by each person. Both similarity and dissimilarity in knowledge are factors that need to be considered in bring people together for successful knowledge collaboration, and the challenge is to strike a right balance between them.

The first challenge to initiate and sustain knowledge collaboration is thus to identify a group of people that shares cognitive proximity. When a knowledge worker seeks collaboration on a problem, he or she first needs to identify the collaboration candidates who have relevant expertise or similar interests in the problem. Research on creating the awareness of other's expertise and locating experts has thrived recently. Systems introduced in Section 3.2 have mainly focused on this proximity of knowledge collaboration in software development.

4.2 Structural proximity

A person's social network constitutes a valuable source of knowledge benefits, in other words, "what you know" is heavily affected by "who you know"[22]. As the research on intensional network [23] has shown that the connectivity with potential collaborators has become one of the essential resources in the modern society. The connectivity of the knowledge workers is reflected in the structural proximity that enables access to the knowledge of each other and participation to the knowledge collaboration process. The connectivity includes the reachability of a knowledge worker to others who are within cognitive proximity and the accessibility to the knowledge held by others. Knowledge collaboration

about a problem is achieved by activating a portion of a social network that connects people of similar interest in that problem, and the connectivity facilitates and coordinates the exchange and combination of complementary elements of knowledge held by a variety of knowledge workers.

Structural proximity can manifest in different forms. Formally institutionalized organizations create one form of structural proximity by subjecting all of members to a prearranged relationship. Connectivity among organizational members is guaranteed by the control and coordination mechanisms inherent in the organization. An apparent shortcoming of such structural proximity to knowledge collaboration is its rigidity in its structure and its inability of creating knowledge flow outside of fixed organizational channels.

Other social forms that provides the structural proximity in knowledge collaboration is further discussed in Section 5.

Reachability of and accessibility to knowledge are the key benefits afforded by structural proximity. Information technology has engenders new forms of structural proximity by enhancing reachability and accessibility with new communication, coordination and collaboration tools. Although research has shown that collocatedness still provides the best mechanism for achieving structural proximity [24], with the advance of modern communication tools such as telephone, fax, email, WWW, and instant messaging, the spatial distance has become increasingly a less important factor in measuring the connectivity of people. Structural proximity can still be achieved even two knowledge workers are distantly located, and the challenge is to explore structural proximity afforded by new communication tools.

Structural proximity for knowledge collaboration is determined by the availability of communication channels that have been established previously, and the availability of the communication medium that can afford timely and sufficient knowledge exchanges. Recognizing the important role of structural proximity in supporting knowledge collaboration, research on modeling and cultivating egocentric social network has flourished recently. Subsequent to their research in intensional network, Nardi et al.[23] have created the ContactMap system that presents a visual representation of a person's social network by analyzing his or her emails. Social Network Fragments is another interactive visualization tool that allows users to navigate the social network patterns in their email conversation [25].

4.3 Relational proximity

Whereas structural proximity refers to the social ties and communication medium that exist between collaborating knowledge workers, relational proximity refers to the attributes of the ties and the relational

closeness that knowledge workers feel toward each other. Cross and Borgatti [26] has found that the existence of arduous relations between a knowledge source and a knowledge recipient is one of the important barriers to knowledge transfer. Research by Nahapiet and Ghoshal [27] shows that the willingness to be engaged in knowledge collaboration is determined by the social relationship. Cognitive proximity presumes the necessity of knowledge collaboration for solving a problem, defining mainly what is shared; structural proximity provides the communication means and medium for possible participation in knowledge collaboration, concerning mainly how is shared; relational proximity motivates the participation in and lubricates the process of knowledge collaboration, explaining mainly why people share.

Nahapiet and Ghoshal [27] characterizes relational proximity along four interrelated aspects: obligations and expectations, trust, identification, as well as norms. Obligations and expectations are the central notions of the relational proximity. As defined by Coleman [28], "if A does something for B and trusts B to reciprocate in the future, this establishes an expectation in A and an obligation on the part of B." Obligations and expectations operate like "credit slips" held among people. If a person holds a lot of credit slips, he or she is more likely to draw those debtors into collaboration, as they need to fulfill their obligations. Such a direct creditor-debtor relationship between individual members provides a strong motivation to participation in knowledge collaboration.

Regular fulfillments of obligations and expectations help the development of trust, which is the belief that favorable past interactions entails positive future interactions. Trust in that other people will reciprocate in the future encourages members to do their own share of contributions as a long-term investment in their relationships. Trust also offers psychological safety in knowledge collaboration because knowledge workers are more willing to admit a lack of knowledge and to risk of offering novel and ill-formed ideas in the interactions [26], and therefore increases the motivation for participation.

As an expectation of continued interaction in the future is important to maintain trust, identification with and mutual identification within a social group is another important sources to develop trust. Identification with a social group means that a person shares the value inherent in the group, and mutual identification means a person takes the values and standards of other individuals in the group as a comparative frame of reference [29]. Due to the inter-dependency of identification, an act of breaking trust with one member will harm the actor's relationship with all other members, having a bad impact on his or her reputation in the social group and might eventually lose

his or her identification with the group as a result of isolation.

When the fulfillments of obligations and expectations occur regularly in a social group and such acts become visible to all group members, social norms such as generalized reciprocity develops. Coleman[30] defines social norm as the socially defined right to control an action held not by the actor but by others. The existence of social norms enables the obligation and expectation relationship extends to the members who are not directly involved. In other words, if the group has seen that A does something for B, A can have a reasonable expectation that someone in the group will do something for him or her, and the group has a reasonable expectation that B will do something for other members. Explicit recognition of group members' past participations is an effective mechanism of motivating participation in knowledge collaboration [31].

Computer scientists who have tried to create knowledge collaboration support systems have paid less attention in taking into full consideration the role of the relational proximity plays partly due to the difficulty in modeling and quantifying relational attributions of social connection. As we can see from the above analysis, interpersonal obligation and expectation is at the center of relational proximity, we believe that tracking the obligation and expectation is the first step toward the exploration of the role of relational proximity in facilitating knowledge collaboration.

5. Social Forms in Support of Knowledge Collaboration

In addition to the formal institutions that provides structural proximity for knowledge collaboration, other less formal social forms that structure knowledge collaboration exist, and have been articulated by a number of researchers. This section first briefly describe the conceptual framework of Dynamic Community that we have proposed to support knowledge collaboration in software development (for details on Dynamic Community, please see[32]), and then compare it with other existing social forms to point out the void that the Dynamic Community theory fills in the spectrum of supporting socio-technical platforms for sustained knowledge collaboration by taking into consideration of all the three proximities required for successful and sustainable knowledge collaboration.

5.1 Dynamic Community

A dynamic community, or DynC, is dynamically formed in a knowledge work space. A knowledge work place consists of a group of knowledge workers, and the knowledge that the workers hold. The group can either be

a formal organization such as a company or an informal online community. A DynC is a subgroup of knowledge workers that forms ad hoc in support of a particular user and a particular task, and disassembles as the task is finished. Unlike a static community, which forms around a particular domain and exists for a long time, a dynamic community forms for a particular task, and exists only for a short period. The members of a DynC are selected using the following two criteria:

- (1) they have expertise on the particular task, and
- (2) they already have social contacts with the particular user.

The first criterion is grounded in the observation that in today's highly specialized world expertise is no more an absolute attribute of person but a relative function of a person and a task. In other words, cognitive proximity can only determined after the task is defined as mentioned before. The second criterion is grounded in the assumption that existing social contacts between the helpers and the helpee could provide extra motivation to engagement in knowledge collaboration based on the social norm of generalized reciprocity [5]. It is meant to take the relational proximity into consideration when we try to form a knowledge collaboration group. The formed DynC provides the means for cognitively proximate and socially proximate knowledge workers to collaborate, namely to create the structural proximity needed for knowledge collaboration.

In summary, a DynC is task-specific: for different tasks, different DynCs will form even for the same user because different groups of experts will be identified; a DynC is user-specific: for different users, different DynCs will form even for the same task because users has different social contacts; and a DynC is in-situ and ephemeral: it is created dynamically when a need for collaboration arose in a real task and disassembles when the task is solved.

5.2 Community of Practice

Lave and Wenger [33] argue that communities of practice are a social form accountable for learning most of the knowledge that professional practitioners own. Learning in communities of practice is situated in direct collaborations with peers. Sustained social relationships and interactions in the community are critical. Lave and Wenger's theory only accounts for the general phenomenon of learning and knowledge collaboration in the setting of a community. A particular learning and knowledge collaboration activity does not involve the whole community, and the learner has to spend a relatively long time to forge personal relationships with other members and carefully select those people to interact with. The people involved in each of such learning activities in fact forms a dynamic community. A dynamic community can be viewed as a community

inside a community of practice, accounting for and supporting for the initiation and unfolding of the learning process in communities of practice. The cognitive proximity in communities of practice is defined at the level of domain, practitioners share a domain is considered cognitively proximate, whereas in dynamic communities, cognitive proximity is defined at a finer level—knowledge involved in a particular task and a particular learning session.

5.3 Community of Interest

Communities of interest are community of communities of practices, bringing together stakeholders from different communities of practices to collaboratively solve a particular problem of common concern that involves knowledge from different domains[34]. Because members of a community of interest come from different domains, their cognitive proximity is distant and the pressing issue is to establish a shared understanding. Relational proximity is a less concern because the common concern of the problem faced by all members is a strong enough motivation, and more importantly, the participation in the communities of interest is often not out of good will only.

5.4 Intensional Network

Intensional networks are egocentric social networks that arise from individuals and their communication and collaboration with other people in workplace activities [22]. When faced with a problem, a knowledge worker activates a subnet of his or her intensional network to solicit collaboration. This activated “live subnet” can be viewed as a dynamic community. An individual’s intensional network is reflected in the relational proximity in dynamic communities. In activating the live subnet of a personal intensional network, the judgment of cognitive proximity is made by the person. Through ethnographic study, Nardi et al. [22] argue that intensional network has become a main form of knowledge collaboration, and knowledge workers actually have difficulties in managing and activating this important social resource. This research underscores the necessity of the conceptual framework of dynamic community and the importance of creating computational systems for its support.

5.5 Knotwork

Engeström, Engeström, and Vähäaho [35] observed that knotwork is another social form for collaboration. Knotworks are the combinations of people, tasks and tools that are unique and of short duration. Typical knotworks include the assembling of airline crews, courts of law, and medical professionals that is driven by situation. Knotworks overlaps with dynamic communities

in terms of the situation driven aspect, short duration, and the repeated tying and untying of the knots. However, two significant differences exist. First, knotworks are structured based on the roles of each member. Although each knotwork has different combinations, the same member always plays the same role. Cognitive proximity depends on the role and is fixed. Second, knotworks are often created by an external force in an institution, not by the members themselves. Therefore, relational proximity is not a concern.

6. Summary

Knowledge collaboration has become an essential activity in the development of modern complicated software system due to not only the required human hands in coding but also the many kinds of knowledge. This paper describes the two basic forms of knowledge collaboration: by transfer and by combination. To support successful knowledge collaboration, three proximities (cognitive, structural and relational) should be taken into consideration when designing computational support. The paper then briefly described the new conceptual framework of dynamic community that we have proposed to support knowledge collaboration in all three proximities, and compared it with existing recognized social forms for the conceptual framework’s uniqueness and application.

Acknowledgements. This research is supported by (1) the National Science Foundation, Grants CCR-0204277; (2) the Ministry of Education, Culture, Sports, Science and Technology of Japan, Grant 15103 of the Open Competition for the Development of Innovative Technology program.

7. References

- [1] J. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally-Distributed Software Development," *IEEE Transactions on Software Engineering*, vol. 29, pp. 1-14, 2003.
- [2] J. Hollan, E. Hutchins, and D. Kirsch, "Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research," in *Human-Computer Interaction in the New Millennium*, J. M. Carroll, Ed. New York: ACM Press, 2001, pp. 75-94.
- [3] W. J. Clancey, *Situated Cognition: On Human Knowledge and Computer Representations*. Cambridge, UK: Cambridge University Press, 1997.
- [4] W. J. Orlikowski, "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing," *Organization Science*, vol. 13, pp. 249-273, 2002.
- [5] M. Polanyi, *The Tacit Dimension*. Garden City, NY: Doubleday, 1966.

- [6] A. Giddens, *The Constitution of Society: Outline of the Theory of Structure*. Berkeley, CA: University of California Press, 1984.
- [7] D. A. Schön, *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books, 1983.
- [8] H. R. Maturana and F. J. Varela, *The Tree of Knowledge: The Biological Roots of Human Understanding*. Boston, MA: Shambhala Publications, 1998.
- [9] J. Lave, *Cognition in Practice*. Cambridge, UK: Cambridge University Press, 1998.
- [10] E. Hutchins, *Cognition in the Wild*. Cambridge, MA: The MIT Press, 1994.
- [11] W. J. Orlikowski, "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing," *Organization Science*, vol. 13, pp. 249-273, 2002.
- [12] L. A. Suchman, *Plans and Situated Actions*. Cambridge, UK: Cambridge University Press, 1987.
- [13] P. Moran and S. Ghoshal, "Value Creation by Firms," in *Academy of Management Best Paper Proceedings*, J. B. Keys and L. N. Dosier, Eds., 1996, pp. 41-45.
- [14] G. Fischer, "A Critic for LISP," presented at Proceedings of the 10th International Joint Conference on Artificial Intelligence, Los Altos, CA, 1987.
- [15] J. E. Robbins and D. F. Redmiles, "Software Architecture Critics in the Argo Design Environment," *Knowledge-Based Systems*, vol. 11, pp. 47-60, 1998.
- [16] Y. Ye and G. Fischer, "Reuse-Conducive Development Environments," *Automated Software Engineering*, vol. 12, pp. 199-235, 2005.
- [17] M. S. Ackerman and T. W. Malone, "Answer Garden: A Tool for Growing Organizational Memory," in *Proceedings of the ACM Conference on Office Information Systems*. Cambridge MA, 1990, pp. 31-39.
- [18] L. G. Terveen, P. G. Selfridge, and M. D. Long, "Living Design Memory: Framework, Implementation, Lessons Learned," *Human-Computer Interaction*, vol. 10, pp. 1-37, 1995.
- [19] A. Vivacqua and H. Lieberman, "Agents to Assist in Finding Help," in *Proceedings of the 2000 Conference on Human Factors in Computing Systems (The Hague, Netherlands)*. The Hague, Netherlands: ACM Press, 2000, pp. 65-72.
- [20] A. Mockus and J. Herbsleb, "Expertise browser: A quantitative approach to identifying expertise," presented at 2002 International Conference on Software Engineering, Orlando, Florida, 2002.
- [21] W. M. Cohen and D. A. Levinthal, "Absorptive Capacity: A New Perspective on Learning and Innovation," *Administrative Science Quarterly*, vol. 35, pp. 128-152, 1990.
- [22] B. A. Nardi, S. Whittaker, and H. Schwarz, "It's Not What You Know, It's Who You Know: Work in the Information Age," *First-Monday: Peer-reviewed Journal on the Internet*, vol. 5, 2000.
- [23] B. A. Nardi, S. Whittaker, E. Issacs, M. Creech, J. Johnson, and J. Hainsworth, "Integrating Communication and Information through ContactMap," *Communications of the ACM*, vol. 45, pp. 89-95, 2002.
- [24] G. M. Olson and J. S. Olson, "Distance Matters," *Human-Computer Interaction*, vol. 15, pp. 139-178, 2001.
- [25] D. Boyd and J. Potter, "Social Network Fragments: An Interactive Tool for Exploring Digital Social Connections," presented at Proceedings of the SIGGRAPH 2003 Conference on Sketches and Applications, San Diego, CA, 2003.
- [26] R. Cross and S. P. Borgatti, "The Ties That Share: Relational Characteristics That Facilitate Information Seeking," in *Social Capital and Information Technology*, M. Huysman and V. Wulf, Eds. Cambridge, MA: The MIT Press, 2004, pp. 137-161.
- [27] J. Nahapiet and S. Ghoshal, "Social Capital, Intellectual Capital, and the Organizational Advantage," *Academy of Management Review*, vol. 23, pp. 242-266, 1998.
- [28] J. C. Coleman, "Social Capital in the Creation of Human Capital," *American Journal of Sociology*, vol. 94, pp. S95-S120, 1988.
- [29] E. Wenger, *Communities of Practice — Learning, Meaning, and Identity*. Cambridge, England: Cambridge University Press, 1998.
- [30] J. C. Coleman, *Foundations of Social Theory*. Cambridge, MA: Belknap Press of Harvard University Press, 1990.
- [31] G. Fischer, E. Scharff, and Y. Ye, "Fostering Social Creativity by Increasing Social Capital," in *Social Capital*, M. Huysman and V. Wulf, Eds., 2004, pp. (in press).
- [32] Y. Ye, Y. Yamamoto, and K. Kishida, "Dynamic Community: A New Conceptual Framework for Supporting Knowledge Collaboration in Software Development," presented at Proceedings of 11th Asia-Pacific Software Engineering Conference (APSEC 2004), Busan, Korea, 2004.
- [33] J. Lave and E. Wenger, *Situated Learning: Legitimate Peripheral Participation*. Cambridge, UK: Cambridge University Press, 1991.
- [34] G. Fischer, "Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems," presented at Proceedings of 24th Annual Information Systems Research Seminar In Scandinavia (IRIS'24), Ulvik, Norway, 2001.
- [35] Y. Engeström, R. Engeström, and T. Vähäaho, "When the Center Does Not Hold: The Importance of Knotworking," in *Activity Theory and Social Practice: Cultural-Historical Approaches*, S. Chaiklin, M. Hedegaard, and U. J. Jensen, Eds. Aarhus, Denmark: Aarhus University Press, 1999, pp. 345-374.