SOFTWARE DEVELOPERS' ONLINE CHAT AS AN INTRA-FIRM MECHANISM FOR SHARING EPHEMERAL KNOWLEDGE

Completed Research Paper

Antti Salovaara and Virpi Kristiina Tuunainen
Aalto University School of Business
P.O. Box 21220, 00076 Aalto, Finland
antti.salovaara@aalto.fi, virpi.tuunainen@aalto.fi

Abstract

Knowledge sharing is crucial for companies in knowledge-intensive domains, but poses challenges for firms operating in a project-based manner. One domain facing this challenge is agile software business, which is based on autonomous teams and where the technological progress poses a constant need to update programmers' knowledge. One solution to the challenge is to use various social media tools, platforms and electronic tools for knowledge sharing. We present a case study on a 150-person software company where so-called frontend developers have adopted Skype chat as their favored knowledge-sharing medium. Analysis of almost 16 000 chat messages from a 17-month period, with 4 749 messages analyzed on a content level, shows several uses of this simple medium for sharing ephemeral programming-related knowledge that becomes quickly outdated. Our findings contribute to a better understanding of online chat as a knowledge-sharing tool and provide implications for research on knowledge sharing frameworks.

Keywords: Knowledge sharing, computer-mediated communication (CMC), software development, Skype chat, ephemeral knowledge

Introduction

One of the biggest management challenges in a knowledge-intensive business domain is the need to maintain and develop the organization's knowledge practices and knowledge base of relevant information. In successful organizations, project teams, organizational units, and individual knowledge workers are motivated and have a possibility to invest in updating their skills, work tools, conceptual understanding and other knowledge-laden assets. Doing so will improve organization's work practices, performance, and significance of its outcomes, as well as employee motivation, among others (e.g., Bhatt 2001).

One means to keep the organization up to date is to leverage the expertise of each member of the organization. By cultivating members in their areas of expertise, creating resources and infrastructure for knowledge sharing, and making sharing an everyday practice in the organization, the organizations will improve the requisite variety as well as redundancy (i.e., beneficial knowledge overlap; Nonaka and Takeuchi 1995) of information within the team, both of which are necessary conditions of favorable culture and climate for knowledge creation, knowledge sharing and innovation (ibid.).

Software Development Project Work: A Challenge to Knowledge Sharing

Knowledge sharing in software development is continuously challenged by the increasingly project-oriented mode of working. In most projects, multi-disciplinarity is necessary (Siau et al 2007), and teams are created anew for each project. As a result, experts have reduced opportunities for knowledge sharing among their team members after a project is completed, as they will soon be assigned to new projects consisting of new team members. In the worst case, workers need to maintain their competence solely by themselves (Boh 2007; Hobday 2000; Ruuska and Vartiainen 2005). Several technological solutions have been developed to cope with this knowledge sharing challenge. Traditional solutions have included knowledge repositories into which knowledge is entered in a codified form for later retrieval and application (Hackbarth and Grover 1999).

In practice, many organizations have adopted common computer-mediated communication (CMC) technologies also for the purpose of technology-mediated knowledge sharing. These commonly complement the non-technological means, such as, team meetings, special interest groups, and other face-to-face gatherings. In a case study of a small 50-person company, Turner et al. (2010) showed that most often the different CMC technologies (i.e., email, chat/instant messaging, phone calls, social network sites, blogs etc.) are used in parallel, supporting each other.

Agile software development is a context that specifically exemplifies the challenge of knowledge sharing in a project-based work. Agile programming is an example of lean management and refers to a managerial practice that supports iterative and incremental software development and self-organizing teams[1]. Organizations practicing agile programming are typically exceedingly project-oriented. They allocate their employees to projects and strive to ensure that all "nuisance factors" hindering productive work are minimized during "sprints" – short bursts of intensive development work. At the same time, the pace of technological change in software business is probably more rapid than in any other domain. Solutions that are state-of-the-art today may be regarded as suboptimal only a year later. Software developers need to update their knowledge and develop new skills on a continuous basis for which, however, very little time may be allocated.

These challenges have several implications for successful intra-firm knowledge sharing among software developers:

- *Knowledge is ephemeral*. Building permanent repositories of knowledge (e.g., enterprise wikis) is perceived obsolete because information becomes outdated very quickly. Lightweight communication methods and tools that allow for synchronous communication are therefore, generally speaking, more suitable for knowledge sharing.
- *External knowledge sources are essential*. The global software business community is extremely well

[1] See, e.g., http://agilemanifesto.org/ (accessed 7 August, 2013).

connected through various Internet sites[2] that distribute programming-related news. Information is shared openly between programmers and, in fact, the different sites compete for audience instead of trying to keep the information closed or proprietary.

- *Knowing the experts in each area is a significant accelerator*. Programming is an activity in which even the most experienced programmers feel "being stuck" quite often. When this happens, finding no help from the Internet, knowing people who to ask for help is an asset (Nardi et al. 2000b).

Derived from the discussion above, our goal in this paper is to analyze how synchronous CMC can be used to support intra-organizational knowledge sharing among software developers. Our empirical data is from a 150-person software company where Skype chat is actively used for knowledge sharing among a subset of programmers working in different teams. Being aware that the chat tool has been used specifically for knowledge sharing with very little casual conversation, its popularity has motivated us to examine the content that is being shared through it. In this paper, we will investigate the knowledge sharing mechanisms utilized in the programmers' work, and in particular, how Skype chat can be used to support these mechanisms. Our two research questions are as follows:

1. What functions can synchronous CMC tools (such as Skype chat) support in knowledge sharing in software developers' work?
2. What roles do synchronous CMC tools acquire in relation to other knowledge sharing mechanisms applied in the company and presented in the literature?

In answering to the first question, we will provide a data-driven bottom-up categorization of Skype's functions as a knowledge-sharing tool in our case company. The categorization includes also a test that verifies the direction of knowledge flow from experts to their less experienced peers. The second question contextualizes the observed Skype chat use in two ways: with respect to the other mechanisms for knowledge sharing in our case company, and the body of knowledge sharing mechanisms addressed in literature. This allows us to better evaluate the importance of synchronous CMC tools in knowledge work generally. To our knowledge, this is the first study to look at chat-based knowledge sharing at a content level. Implications of the findings are addressed in the end of the paper.

Related Research

Successful knowledge sharing depends on several factors, including sharing-conducive organizational culture, infrastructural support, worker motivation, and location and scheduling arrangements that allow interpersonal communication (e.g., Davenport and Prusak 1998; Hendriks 1999; Tampoe 1996). In favorable conditions, workers are able to develop *expert communities* and groups that support influx of knowledge both within an organization and with external knowledge sources (Nardi et al. 2000b; Wenger 1998). These flows may form complex patterns. In software development and in other domains where the pace of change is fast, and there are no "old timers" who would possess all the relevant knowledge in the domain and distribute it unidirectionally to others in the network (cf. Hakkarainen et al. 2004). Instead, the flow may take place in any direction between the community members. The central aspect here is the networks' temporary nature: they are created opportunistically, being ego-centric in the sense that every expert maintains a slightly different network of people who serve as knowledge sources and collaborators (Nardi et al. 2000b). In large endeavors, such as open source development, getting in touch with the most knowledgeable experts requires strategic interaction, such as selectiveness on who to communicate with (Kuk 2006).

Networks may change rapidly in project-based work. When a project finishes, its team is often disbanded and its members are recruited to other projects and, consequently, into interaction with new people. Teams that succeed in coordinating their expertise in such circumstances achieve higher levels of performance than others (Faraj and Sproull 2000).

[2] Each programming language, environment, framework, and library has its own forums, Twitter feeds, blogs and content aggregators that distribute timely information. For a review of mechanisms for staying up to date on frontend development, see, e.g. www.smashingmagazine.com/2012/08/09/productivity-staying-up-to-date/ (accessed 7 August, 2013).

It is also possible that experts of a certain community never work together in the same projects. Markus (2001) has described such *shared-work practitioners* – that is, people doing similar work in different settings or teams – as another important organizational group. When working separately, these people may not originally form a community, but over time, they might eventually recognize a need for mutual knowledge exchange and seek opportunities for doing so. These are the people whose knowledge sharing practices are considered in this paper.

Knowledge Sharing as Storage and Retrieval

Knowledge sharing through personal connections, without organizational support, has been found to have a limit. In organizations with 150 workers or more, effective knowledge sharing on a personal level becomes problematic (Serenko et al. 2007). At that level, workers start to find it difficult to know each other's areas of expertise. To address this issue, organizations use information and communication technology (ICT) enabled knowledge sharing systems to effectively combine and utilize knowledge resources that are distributed amongst the employees and groups (Boh 2007). Knowledge management systems may be targeted at supporting knowledge creation, storage, sharing, or application of knowledge in organizations (Alavi and Leidner 2001).

Traditional information systems (IS) based solutions for knowledge sharing include storage and retrieval systems in which documentation, marketing data, instructions and other relevant knowledge are stored for later retrieval. Their purpose is to capture the knowledge and expertise of the workers and make it accessible for others. The underlying goal of these repositories is to externalize and explicate knowledge that otherwise would remain tacitly possessed by individuals in an organization (Nonaka and Takeuchi 1995). One suggestion for achieving this involves elicitation of knowledge in a narrative format (e.g., Davenport and Prusak 1998; Linde 2000). However, this work can become prohibitively expensive. Also, the knowledge base must be updated continuously in order to remain relevant, making the effort excessively time consuming in swiftly advancing environments such as software development.

As a result, no widely accepted IS-based solutions for large-scale knowledge management – or sharing more specifically – have been proposed. A less rigid approach based on free-text search and tagging has emerged during the 2000s. This solution bypasses dedicated codification and classification efforts and relies more on employees' ability to carry out successful queries into textual data (e.g., Grudin 2006).

Knowledge Sharing as Communication

In the literature reviewed above, knowledge sharing has been treated mostly in the context of creation, searching, and reading of document collections. However, a significant amount of knowledge sharing and transfer takes place through interpersonal communication, such as talking, working together, and exchanging messages. We review below the findings related to CMC-based sharing, with a particular focus on practices based on chat and instant messaging (IM) tools.

Several different communication systems are often used simultaneously in organizations, with email being adopted by virtually all members, and other online tools (IM, Twitter, wikis, weblogs, social networking sites such as Yammer, and virtual worlds) as well as their combinations having their proponents (Turner et al. 2010). In software development – the domain of interest in this study – communication with peers is a central part of the work. Of all the working time in programmer's work, over half of the time may be spent interacting with coworkers, with a purpose of serving different information needs (Ko et al. 2007). Even if the media richness theory (e.g., Daft et al. 1987) would suggest that a multimodal (e.g., video-based) tool or channel would better support expert-level communication, text-based communication is the primary means of communication (e.g., Gutwin et al. 2004; Lee 1994; Rasters et al. 2002). Two communication situations stand out: when developers want to coordinate development activities (*coordination communication*) and when they want to acquire knowledge (*expertise communication*; see Nakakoji et al. 2010).

The popularity of IM/chat as a communication tool has not gone unnoticed by researchers. Its popularity in non-work use is at least partly due to its perceived usefulness and fun, as well as its ability to fulfill one's need for attachment and allow one's friends, family members, and others to gather online (Lou et al. 2005). Several studies have also addressed these tools' use in organizational contexts. Pazos et al. (2013)

found significantly greater use of IM for collaboration tasks (e.g., seeking for help in identifying a problem with a system bug) than for so-called cognitive conflict tasks (e.g., discussing business requirements for a new system). In content classification studies, workers have been found to engage in making quick questions and asking for clarifications, coordination and scheduling work tasks, coordinating impromptu social meetings, and keeping in touch with friends and family (Isaacs et al. 2002; Nardi et al. 2000a). Several of these functions may co-exist in the same exchanges. For example, in one study, on a level of individual messages, software development team members communicated mainly about work (69%) and availability (13%), while greetings (7%), humor (5%), non-work topics (3%), and other issues (4%) were conversed less frequently (Handel and Herbsleb 2002).

Although some researchers have theorized that the use of IM decreases productivity (Rennecker and Godwin 2003), others have found that employees using IM communications can work productively on normal tasks, although they may take longer to complete them (Mansi and Levy 2013). With respect to the choice between IM and email, the effect on performance may depend on the way in which the teams' work has been organized: IM is used as a replacement for face-to-face discussion while email seems to be more suitable for team-wide, inter-site communication (Niinimäki 2011).

Knowledge Sharing Mechanisms

On a higher level, storage and retrieval as well as communication are examples of different *knowledge sharing mechanisms*. As discussed earlier, software development is largely project-based work. To enable effective sharing of knowledge across projects, individuals embark on knowledge-sharing mechanisms as a means to access knowledge and information from other projects (Boh 2007).

Knowledge sharing can be typified along several dimensions. For example, sharing may take place tacitly (e.g., through observation and imitation) or explicitly (e.g., through reading documentation); unidirectionally or multidirectionally; in an ad hoc or institutionalized manner; on a personal level or by addressing large amounts of people simultaneously; or, through rich or weak medium (e.g., Alavi and Leidner 2001; Boh 2007). Different *knowledge sharing frameworks* combine two or more of these dimensions for the purpose of generating a more comprehensive understanding of sharing practices and opportunities. They differ from typologies of knowledge by describing forms and methods of sharing knowledge instead of analyzing the epistemological nature of knowledge itself.

Two knowledge sharing frameworks are applied in this paper. Boh's (2007) framework extends the findings of earlier research (e.g., Hansen et al. 1999) that has established that organizations facilitate sharing of knowledge between individuals by using codification or personalization mechanisms. These depict mechanisms for sharing explicit knowledge or tacit knowledge, respectively. Codified knowledge is carefully articulated, captured and stored in documents and databases for others to access and use (Boh and Wong 2013). Personalization, in turn, emphasizes personal reflection and interaction (Storey and Kahn 2010). ICT's role in personalization concerns knowledge transfer through direct person-to-person contact (Yu et al. 2010).

Boh's (2007) addition to this distinction between codification and personalization is another dimension to categorize whether the mechanisms are informal (individualized) or formal (institutionalized). This dimension has its roots in the theory of organizational socialization (Van Maanen and Schein 1979). Informal and formal knowledge sharing mechanisms distinguish individual-level sharing from collective-level sharing (Boh and Wong 2013). *Informal* mechanisms are ad-hoc and unstructured, and support knowledge sharing in an unplanned manner. *Formal* mechanisms, in turn, are structured, and support the transference of knowledge from one to many individuals by utilizing the organizational structures and routines (Boh 2007; Boh and Wong 2013).

According to Boh's (2007) framework, knowledge sharing through *informal personalization* takes place at an individual level in an ad hoc and informal manner, through various informal channels. Technologies that typically support this mechanism include email and IM. When supported with an organization-wide database that has good search capabilities, documents and other project artifacts can be shared with *informal codification* mechanisms, also at an individual level and in an ad hoc manner, such as through document exchange with e-mail. *Formal codification* mechanisms, in turn, are institutionalized and have a strong emphasis on the use of ICT to create electronic repositories (e.g., using organizational intranet; cf. Boh and Wong 2013) for storage, search and retrieval. Finally, *formal personalization* mechanisms are also institutionalized in the routines and structure of the organization but are more collective in the form of joint exercises, special interest groups, and other expert meetings (Boh and Wong 2013). Boh's (2007) framework can be understood as a typology of *modes* by which knowledge is shared: whether it takes place a personal level, collectively, tacitly, or in a codified form, in different combinations.

The other framework useful for the purposes of our study is the one developed by Berends, van der Bij, Debackere, and Weggeman (2006). Instead of categorizing modes, it focuses on the origination of shared knowledge, such as who shares it, to whom it is targeted, and whether it is intended to solve a particular problem or not. The three dimensions of this framework are the novelty of content (new vs. existing or retrieved from external source), the actor who determines that knowledge needs to be shared (the sharing person, another person who needs the knowledge, or management), and the intended orientation for knowledge (one's own problem, other person's problem, a problem shared by the sharing and the receiving person, or no particular problem orientation). Based on these dimensions and a list of altogether 24 logically distinct knowledge sharing mechanisms, Berends et al.'s study (2006) identified seven most frequent mechanisms: 1) diffusion (sharing existing knowledge for others without an imminent problem in mind); 2) pushing (sharing existing knowledge in order to help in another person's particular problem); 3) information pooling (sharing existing knowledge to aid in a shared problem); 4) information retrieval (asking for knowledge from someone, with a particular problem in mind); 5) thinking along (coming up and sharing one's ideas related to another person's problem); 6) self-suggestion (sharing one's thoughts on a particular problem, thus helping oneself in formulating new knowledge); and 7) collaborative problem solving (creating new knowledge related to a shared problem by sharing knowledge together). Berends et al. (2006) suggest that their framework can be used for more coordinated analysis of knowledge sharing observations. Organizations can also assess their practices by comparing their processes against this framework.

Our case study analyses the extent of an online chat tool's knowledge sharing capabilities through the lenses of these frameworks. Existing research has not attempted, to our best knowledge, to create a conceptual understanding of a synchronous CMC tool's role in sharing of ephemeral knowledge. That knowledge may be ephemeral and has implications to its sharing has been only fleetingly mentioned in extant literature. An early remark has been made in Siemieniuch and Sinclair's (1999) initial work on knowledge lifecycles and a "half-life of knowledge", but their focus has been on organizational processes on a management level. Alavi and Leidner's review (2001) only mentions it in a remark on the "ephemeral nature of some knowledge" (p. 112) without further analysis. Leseure and Brookes (2004) note project-related knowledge as an example of ephemeral knowledge due to its obsolescence after project completion. They divide knowledge into kernel and ephemeral kinds and discuss the structural elements of each in organizational information but do not analyze sharing of knowledge further (Leseure and Brookes 2004). Knowledge workers' practices of sharing ephemeral knowledge have therefore remained an unaddressed topic. In the following section, we describe a study that is the first attempt to address this heretofore-open issue.

Empirical Study

In the introduction, we presented project-based work in agile software development as a domain with particularly evident challenges for knowledge sharing. To better understand these challenges and the ways in which companies may cope with them, we conducted a case study in a 150-person software company, hereafter referred to as Delta. Delta is headquartered in Finland with offices in five cities, two of them abroad. Delta subscribes to agile programming and lean management. Its business model is based on commissioned work: its products are tailor-made software products (websites, web applications, mobile services etc.). Its customers come from different industries and are of different sizes, ranging from small local companies to large international ones. Building on the dimensions outlined in the above-presented framework of knowledge sharing mechanisms (Berends et al. 2006; Boh 2007; Boh and Wong 2013), we analyzed the knowledge sharing practices of Delta software developers. Our general focus was on Delta's knowledge sharing mechanisms while our detailed focus was on ICT-supported mechanisms.

We find Delta a very interesting context for such a study. Being a company subscribing to agile software development, the work at Delta is organized into projects that have tight iterative cycles. The cross-functional project teams have members who represent complementary fields of expertise. People who share the same expertise are mostly scattered into different projects without many naturally occurring possibilities for meeting each other.

Knowledge sharing is increasingly becoming a challenge at Delta as their headcount has been growing at a steady rate every year. With more and more people, it is all the time harder for workers to know who to talk to in each programming-related issue. Delta takes the challenges in knowledge management seriously, and has succeeded in creating a strong pro-sharing culture. It also puts a lot of effort on employee wellbeing, which is evidenced in high ratings in the Great Place to Work Institute's[3] annual audits through several consecutive years.

Frontend developers comprise one of the above-mentioned scattered groups. These experts are needed in virtually all of Delta's software development projects. They are responsible for programming the user interface related parts as well as those gateways with which the different devices (computers, phones, tablets) communicate with the backend servers.

To alleviate knowledge sharing problems, a dedicated Skype chat tool is in an active daily use among frontend developers and gathers dozens of new messages every day. This suggests that online chat tool is of considerable utility for its users. We were granted a permission to analyze this communication. When familiarizing ourselves with this corpus, we were surprised by the richness and depth of information that had been conveyed in such a simple text-based tool.

Our main interest was in the extent that Skype chat is utilized in both informal and formal knowledge sharing and whether it is used to support both personalization and codification mechanisms or different origination mechanisms of knowledge sharing. We were also interested in the chat tool's role in knowledge sharing with respect to other possible mechanisms among the frontend developers. At Delta, such mechanisms include frontend developers' Friday meetings and a company-wide enterprise social media service.

We will next describe the data collection in more detail, and offer a general overview of the knowledge sharing practices at Delta, after which we present the results of the study. While our primary focus is on the Skype chat tool, we will also address the knowledge sharing mechanisms utilized by the developers.

Data Collection and Preparation for Analysis

The Skype chat corpus we retrieved from Delta consists of communication exchanged between September 2011 and December 2012 in the frontend developers' online chat channel, totaling 15 804 messages. The dataset did not include other forms of communication between programmers, such as their one-to-one chat conversations or other active group chats at Delta. Also, it did not include the first four months (from April to August 2011) of communication within their chat channel but was otherwise complete. During the initial examination of the Skype chat content we observed that it would be easy to segment into separate conversations. Members of the chat discussed in the chat in a focused manner, rarely leading the topics of conversation into new issues. There were also silent periods in the channel, usually marking changes to new topics of discussion. This motivated our approach to use conversations as the units of analysis.

The subsequent segmentation and classification of the rather technically oriented discussion in the chat was considerably helped by the first author's 3-year side-job experience as a web interface programmer in a small startup company. The first and last three months of the data (2 441 and 2 308 messages) were segmented into 401 (146 + 255) separate conversations. Our decision to focus on these time periods was based on a wish to include data from both ends of our dataset (to increase natural variance) and from the same time of the year (to exclude possible artifact effects arising from comparing holidays with work-intensive periods, for instance). Chat content was examined in a chronological order, paying attention to the time intervals between successive messages. These and the actual message contents were used as a basis for deciding whether a message continued the most recent conversation, opened a new one, or returned back to an earlier conversation. Often a new conversation was also marked with a salutation (e.g., "hi guys"), which served as an additional cue helping in the task. In the analysis of the first three months of the data, segmentation was a separate stage and content classification (which will be presented in the

[3] See www.greatplacetowork.com. In 2012, the European evaluation covered 15 countries and over 1 500 organizations, making it possibly the largest audit of organizational culture in Europe.

next section) followed after it. In the analysis of the last three months of data, the two stages were carried out simultaneously.

To further support the analyses of knowledge sharing, we created a list of recognized experts in frontend development within Delta. We approached three knowledgeable persons in Delta (two active members in the chat and a senior programmer with a long-term experience from several fields of programming, including the frontend). We classified each chat member as an expert if at least two out of three of our informants gave this indication. This procedure singled out six persons from the rest of the 26 people in the 3 + 3 months of segmented data.

Furthermore, the first author visited three frontend developers' weekly Friday meetings and one company-wide monthly meeting, providing us with useful qualitative understanding on the company practices and culture.

What Functions Can Synchronous CMC Tools Support in Knowledge Sharing?

Figure 1 presents the temporal fluctuation in the number of messages on a general level, broken down both by person as well as by month. There are two drops – in 10/2011 and in 07/2012 – in the data. The former proved unexplainable since the chat members could not remember any particular reasons for inactivity. The latter was due to a summer holiday season. Excluding this, the chat proved to be extremely popular. While the five most active members (whose percentages can be found in Figure 1) accounted for 58% of all the messages, 15 members on average (min 8, max 21) contributed 10 or more messages to the discussion each month. During the entire 16-month duration covered in our data, 86 people participated in the chat with at least one message, 32 of which participated in the discussions in our segmented subset.

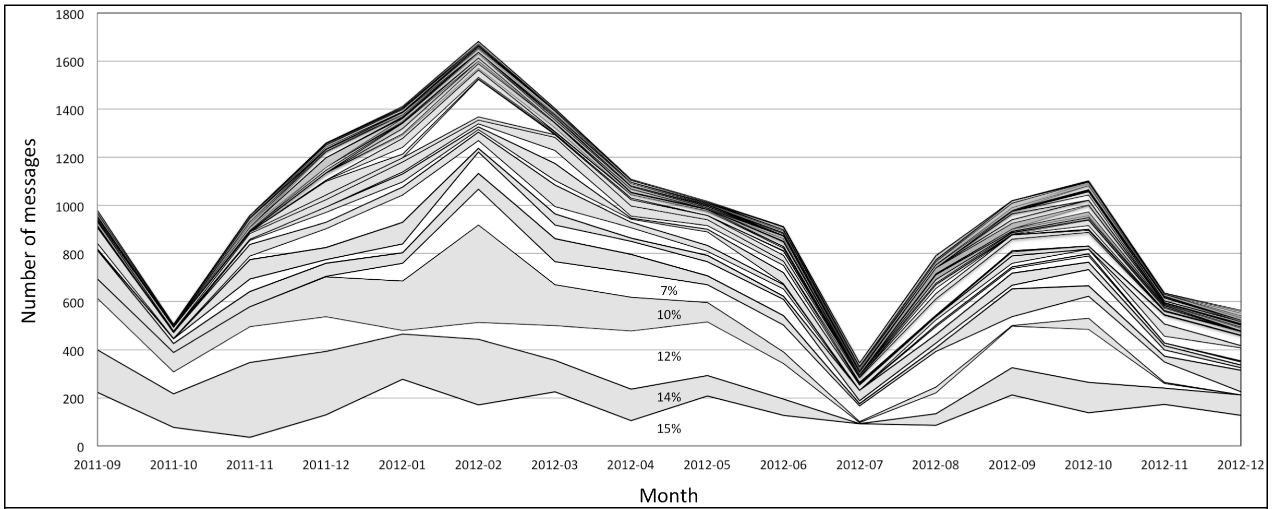

Figure 1. Chat Message Counts per Month and Person

To understand how the chat tool was used among frontend developers, we carried out a content-level qualitative categorization for the subset that we had segmented in our data preparation stage into separate conversations. The process consisted of an open coding phase during which different categories for conversations were explored, followed by a more straightforward classification stage.

During open coding, special attention was given to the ways in which conversations were opened (e.g., whether it was formulated as a question or not) and terminated (e.g., whether a question received a solution or not, or whether the conversation simply "died" or was overridden with a discussion about a new topic). Already after an inspection of 30–40 conversations, it was evident that the openings often determined the format for the conversation that followed (e.g., trouble-shooting, advice-giving, opinion-exchange, joking etc.); that practically all conversation addressed programming-related issues instead of casual chit-chat; and that the members very rarely steered ongoing conversations to new issues. Because of this, the categories developed during open coding could be treated as being mutually exclusive and a rather small number of categories covered all communication in the channel. These observations remained valid through the rest of the data analysis.

Coordination communication (Nakakoji et al. 2010) in terms of discussions on projects' internal coordinative issues (e.g., division of labor, scheduling, etc.) was non-existent in our data. This was to be expected, based on the frontend developers' position in the organizational structure: they constitute a horizontal work group, meaning that each developer works in a different project. Therefore, projects' internal issues are discussed in other forums.

Despite the lack of discussions on coordination and other project-internal matters, the chat communication was highly focused on *expertise communication* (Nakakoji et al. 2010), and interestingly, dominated by explicit knowledge sharing. Table 1 presents a summary of the conversation functions from the first and last three months of our online chat data.

Table 1. Skype Chat Conversations Classified on the Content Level by Their Function

Function	Frequency (number / % of conversations)	Avg. length (number of messages)	Avg. number of participants	1st message is a question (%)
Informing	125 / 31%	7.1	2.4	1
Peer help	119 / 30%	20.1	3.7	96
Remarks on programming detail	43 / 11%	12.2	2.7	9
Comment on/to somebody	20/ 5%	11.8	3.2	25
Joking	13 / 3%	6.1	3.8	15
Other	64 / 16%	9.2	3.1	8
Unclassifiable	17 / 4%	1.9	1.2	0

Informing was the most common function Skype chat was used for. Here the initiator of the discussion offers information, rather than asks for any. The conversations belonged to several different sub-categories. Most commonly, programmers posted web links to pages containing information that they thought their colleagues should be aware of:

Conversation #93 (all 6/6 messages):

KP: cool way to configure a tooltip position:

KP: position: { [line break] my: 'bottom center', [line break] at: 'top center' [line break] }

KP: (for you guys who want to read code like english)

MA: where's that from? [posts Skype's standard smiley]

MA: ... my 'center' at 'top' of my 'bottom'

KP: http://craigsworks.com/projects/qtip2

Another common sub-category of *informing* was news about a module that the poster had recently created and which the others could use in their work as well:

Conversation #35 (all 3/3 messages):

TT: I has made a new meny for my website: [www link to source code in an open repository]

TT: will publish the scroller later as a separate project in github, supports any number of elements [posts Skype's standard smiley]

TT: and try resizing the window too... should adjust the circle radius

The remaining sub-categories in *informing* included news about upcoming programming-related events and links to the news in popular media about programming.

Peer help was the online chat's second most common function. Quantitatively speaking, its main difference from *informing* was the considerably higher number of messages in each conversation. They were usually initiated with requests for help after which lengthy conversations could follow. The following excerpt exemplifies a start of a typical peer help conversation, with a question about an open-source JavaScript programming framework called Backbone and its good and bad sides:

Conversation #66 (only first 5/63 messages):

MA: is it a stupid idea to try to create a base view in Backbone (e.g. header, content, footer) and sub-views that extend from the parent one?

MA: trying to figure out a nice way of only having the content that actually changes

KP: i have a super view Page that all other (pages) extend

KP: that way it's easy e.g. to add an animation to page changes

JR: Yes, extension can be very natural with Backbone.

...

Usually the peer help discussions were about open-ended questions that did not have an easy answer. The reason for open-endedness can be explained in several ways. First, the programmers told us that they exercise *self-screening* for possible solutions, and will post a question to the chat only after having failed to find an answer in the Internet. Simple questions did not appear in the online chat because they could be resolved by consulting question-and-answers forums such as StackOverflow or programming-related weblogs in the Internet. Second, software development is an activity full of so-called *ill-structured problem solving* (Robillard 1999; Simon 1973), that is, programming tasks that can be solved in several different ways, each one with different pros and cons. In such tasks, listening to peers' opinions may save a lot of time. This may have been the second reason for the prevalence of open-ended questions in the chat. Third, the number of different frameworks and issues related to each technological platform and programming environment is so large and keeps on changing so quickly that requesting for others' opinions is a useful way to fill the gaps in one's own knowledge. In these cases open-ended questions about new frameworks are easier to ask than very specific questions.

Remarks on programming details were the third big conversation category. These conversations contained programmers' expressions of frustration about oddities in their present work. Compared to peer help and informing, in some cases these conversations' knowledge-sharing function is unclear, and they may rather have to do with stress-relief and seeking of emotional peer support:

Conversation #84 (only first 6/11 messages):

EK: nice! didn't know this yet another useful trick: ![] == []

JR: ...the f***

EK: [posts Skype's devil smiley]

KP: EK: tell moar! that was only in the comments?

EK: well that will be noted in my book "useful javascript tricks", #421: getting true as a result: ![] == []

KP: [posts Skype's standard smiley] well that's useful

...

However, in 25% of these conversations (i.e., 3% of all), *remarks* initiated a discussion where the poster's colleagues started to help the poster work around the problem, leading to knowledge sharing about a programming languages' or frameworks' working logic:

Conversation #91 (only first 3/61 messages):

MA: gnahh, just can't seem to wrap my head around the backbone event model

MA: how to properly refresh a list of items in a view inside a view so that the event handlers won't go missing when the view gets redrawn

TT: if you are recreating all items, the listeners should be created at the same time

The remaining categories – *specific comments addressed to somebody* (e.g., thanks for someone's help posted afterwards), *joking*, *unclassifiable messages* (e.g., messages posted but later removed by the poster), and *other* conversations (most commonly discussions on who would be willing to give a presentation on an upcoming Friday meeting) – constituted only 28% of all the data and will not be discussed further here.

Summing up the percentages of functions that can be consider to involve knowledge sharing among chat members – *informing* (31%), *peer help* (30%) and those remarks on programming details that led to knowledge sharing (3%) – we get the total of 64%. This is a notably higher share than what has been found in previous classifications of online chat contents in the work context of high-tech company professionals (Isaacs et al. 2002), open-source software developers (Gutwin et al. 2004) and software development teams (Handel and Herbsleb 2002). A possible reason for the difference is that our data was devoid of any actual joint work carried out with the help of the chat tool. Although the chat participants were allocated into separate projects, the popularity of the tool suggests that the knowledge sharing ability offered by it was perceived crucial to frontend developers.

However, a high percentage of knowledge sharing conversations does not in itself prove that the chat would benefit the members optimally. For example, it does not reveal whether it were only novices who participated in the discussions while the experts were too busy attending, or whether the chat was mostly used for "elite talk" between experts. As an additional verification, we examined this question by analyzing the *peer help* conversation category in more detail. We counted how often each chat member initiated a conversation or was the first person to respond. With the help of our list of frontend programming experts obtained from our three informants at Delta, we labeled each initiation and response either as expert-given or less-expert-given.

The cross-tabulation in Table 2 shows how experts and non-experts differed in their frequency of initiating peer help discussions or being the first ones to respond to others' requests. The table provides both the observed values in the data as well as expected frequencies required for statistical analysis calculated from the 100 conversations that had at least 2 participants in the peer help category. Overall, 69.3% of the messages were from experts and 30.7% from non-experts. Based on this information, we calculated the expected frequencies for each quadrant by multiplying the total number of conversations (100) with the corresponding percentages (e.g., expected frequency for expert initiator and non-expert respondent conversations was $0.693 \cdot 0.307 \cdot 100 \approx 21.28$).

Because the observed value (36) in the table's expert–expert cell is lower than its expected frequency (48.01), the data suggests that experts were more active in engaging in helping less-expert programmers than withholding information from them. A Chi-square contingency table analysis gives a significant support for this hypothesis ($\chi^2(1) = 10.12$, $p = .0015$).

Table 2. Frequencies of Conversation Initiations and Responses in the Peer Help Conversation Category, Broken Down by Members' Levels of Expertise (Expected Frequencies in the Parentheses)

	First respondent	
Initiator	**Expert**	**Non-expert**
Expert	36 (48.01)	21 (21.28)
Non-expert	26 (21.28)	17 (9.43)

To conclude, the analyses in this sub-section described the functions of knowledge sharing in our data, reported the high percentage of knowledge sharing related conversations (64% of the total), and verified statistically the direction of knowledge sharing in one the communication functions. In the following sub-section, we will analyze the relation of the Skype chat to other knowledge sharing mechanisms at Delta as well as knowledge sharing frameworks in literature.

What Roles Do Synchronous CMC Tools Acquire in Relation to Other Sharing Mechanisms?

The data-driven observations of knowledge sharing presented in the previous section offer a possibility to examine the value of synchronous CMC tools more generally. To carry out such an analysis, the observations about Skype chat practices need to be contextualized with respect to the other parallel knowledge sharing practices at Delta. We remarked in our description of data collection for this study that Skype chat was but one of the knowledge sharing mechanisms identified among Delta's frontend developers. Namely, although most employees work at their customers' sites, the company recommends its workers to work at the company premises every Friday. The aim of this practice is to support and maintain knowledge sharing between the employees. Given this opportunity to meet face to face (or through a video link), the frontend developers initiated in March 2012 a practice of having *Friday afternoon meetings*. These meetings consist of short informal presentations prepared by different workers, usually dealing with their experiences of programming with novel programming frameworks. Usually a meeting has 2–3 such presentations, each one including an open-ended questions-and-answers session.

In addition to Skype chat, another electronic means for knowledge sharing at Delta is the enterprise social network service *Yammer*[4] that is in use on a company-wide level. While it was used for company-wide matters such as formal bulletins and discussions on official *modus operandi,* one of our informants told that for frontend development related matters its importance was minimal and that many frontend developers did not follow it, at least not actively. In addition, *face-to-face communication* and *emails* are ubiquitous means of communication also at Delta, used for communicating issues both related to and beyond frontend development.

When evaluated along the most commonly mentioned knowledge sharing dimensions – explicit vs. tacit; formal vs. informal; unidirected vs. multidirected; individual vs. collective; and, weak vs. rich medium – the above-presented classification of communication functions presents the Skype chat communication at Delta mostly as *explicit* (due to the textual medium), *informal*, and *multidirected* knowledge sharing. To assess the directedness and richness dimensions is, however, problematic. By addressing individual employees' problems through peer help conversations and generally relevant issues through informing, Skype chat serves both individual and collective functions. Also, its interactivity allows for rich communication even though its text-based channel has a low bandwidth. This attests to the earlier critique (e.g., Lee 1994; Rasters et al. 2002) on the media richness dimension's validity in the context of knowledge sharing.

A more holistic picture on Skype chat's knowledge sharing capacity can be acquired by viewing its communication functions in terms of the frameworks by Berends et al. (2006) and Boh (2007). We performed a mutually exclusive categorization of the 401 conversations in our data into Berends et al.'s (2006) full list of 24 knowledge-sharing mechanisms. Lacking explicit principles for coding the data using this framework, this analysis was explorative. Nevertheless, even with possible errors, the results were unambiguous enough to warrant being reported. We found out that only two knowledge sharing mechanisms singled out in Berends et al.'s (2006) framework could be identified in large quantities in the Skype chat data: *diffusion,* as in proactive sharing of knowledge without being intended to solve anybody's specific problem, and *information retrieval,* as in asking for others' help in one's problems. The results depicted in Table 3 indicate considerable overlaps (95% and 98%) between these mechanisms and the informing and peer help functions in our own analysis. The other conversational functions, on the other hand, were not easily classifiable. Whenever classification was possible (mostly in the remarks and comments functions), the conversations were labeled as diffusion and information retrieval.

Bearing in mind that Berends et al.'s (2006) framework has been specifically developed for classification of knowledge sharing practices, its success in classifying practically all the knowledge sharing related messaging also in Delta's developers' Skype chat is not surprising. However, what appears surprising at first is that the framework suggests Skype chat serving only two types of practices, leaving most of the categorized 24 mechanisms unsupported. The reason however stems from the combinatorial way in which the framework's categories of mechanisms have been postulated. The framework contains several categories for situations where company management would determine the knowledge to be shared or where the tool would be used for teams' internal collaborative work. While Skype would probably be suitable for also such forms of knowledge sharing, that was not part of the frontend developers' practice at Delta. All in all, however, the two categories of knowledge sharing related communication identified in Delta's Skype chat are among the seven most frequent types recognized also in Berends et al.'s (2006)

[4] www.yammer.com

Table 3. Percentages of Skype Chat Conversations, Presented by Function, When Mapped onto Five Knowledge Sharing Mechanisms in Berends et al.'s (2006) Framework

	Knowledge sharing mechanism					
Function	**Diffusion**	**Information retrieval**	**Collaborative problem solving**	**Pushing**	**Self-suggestion**	**Unclassifiable into any mechanism**
Informing	95	-	-	3	-	2
Peer help	-	98	-	1	-	2
Remarks on programming detail	10	15	-	-	3	73
Comment on/to somebody	-	21	-	-	-	79
Joking	-	-	-	-	-	100
Other	2	6	2	-	-	90
Unclassifiable	-	-	-	-	-	100

Note. No instances were found for the remaining 19 mechanisms in Berends et al.'s (2006) framework. These mechanisms have therefore been omitted. To improve readability, the two mechanisms with highest number of instances have been shaded with gray. Also, the zero values have been replaced with a hyphen (-).

study. Bearing this in mind, the Berends et al.'s (2006) framework summarizes the practices at Delta's Skype chat very succinctly, simultaneously pointing out knowledge sharing mechanisms for which Delta's developers have not appropriated this Skype chat and which have been addressed using complementary methods, such as with Yammer and Friday meetings.

Applying Boh's (2007) framework into the same data presents quite a different picture. Table 4 shows how the knowledge sharing mechanisms at Delta can be described in terms of Boh's (2007) two dimensions. Skype chat can be found in three of the four quadrants of the table. Starting from top-left, by resembling face-to-face informal interaction, albeit in a textual form, Skype chat is used to share knowledge through *informal personalization* mechanism. But, because the members in the online chat do not share the same project context, the issues discussed through chat are usually abstracted away from the individual workers' programming matters. Communicating in a manner that is generally understandable is a necessary condition, for example, in *peer help* conversations and joint troubleshooting. In other words, the information needs to be codified into a generally understandable format, and therefore Skype chat also exhibits characteristics of *informal codification*. Communication is codified also in a sense that the chat messages are stored in a history and are in principle retrievable later on using technical terms that have appeared in previous discussions. It is however unclear how much Skype chat is used in this manner at Delta, since Skype chat readily offers only limited options for such a retrieval.

Table 4. Knowledge Sharing Mechanisms (adapted from Boh and Wong 2013) Among Delta's Frontend Developers

	Informal (individualized)	**Formal (institutionalized)**
Personalization	Face-to-face chatting Skype chat	Weekly Friday meetings Yammer Skype chat
Codification	Email Skype chat	

Also formal uses can be noticed in frontend developers' online chat practices. In the *informing* category, the chat tool serves as a *formal personalization* based knowledge sharing mechanism: a lightweight and dynamically updated knowledge base that members can follow in order to become notified of relevant programming-related web links. The fact that such links may not always lead to discussions is a sign of a different kind of knowledge sharing mechanism than is the case with above-described informal mechanisms. The fourth type of knowledge sharing mode – *formal codification* – was not represented in the chat data, since Delta had not institutionalized the ways in which knowledge shared in the online chat discussions (or, for that matter, through any other channels, tools or media, either) would be codified and stored for a later or wider use. This is in line with the ephemeral nature of a large part of software development knowledge that defies the reason to even attempt to build any permanent, digitized knowledge repository.

Also formal uses can be noticed in frontend developers' online chat practices. In the *informing* category, the chat tool serves as a *formal personalization* based knowledge sharing mechanism: a lightweight and dynamically updated knowledge base that members can follow in order to become notified of relevant programming-related web links. The fact that such links may not always lead to discussions is a sign of a different kind of knowledge sharing mechanism than is the case with above-described informal mechanisms. The fourth type of knowledge sharing mode – *formal codification* – was not represented in the chat data, since Delta had not institutionalized the ways in which knowledge shared in the online chat discussions (or, for that matter, through any other channels, tools or media, either) would be codified and stored for a later or wider use. This is in line with the ephemeral nature of a large part of software development knowledge that defies the reason to even attempt to build any permanent, digitized knowledge repository.

While Skype chat appears to represent three out of the four modes of knowledge sharing mechanisms, Yammer and the Friday meetings only represent the *formal personalization* mechanism. Yammer is one of the "official" communication channels and therefore represents a very institutionalized knowledge sharing mechanism. The Friday meetings, on the other hand, are formal in a sense that they also have a number of reoccurring features: the space for the meeting is reserved in advance, the meeting is structured as a series of presentations, and the presenters are recruited during the weekdays leading to Friday.

The fact that Skype chat supports both informal and formal, as well as personalized and codified knowledge sharing (and is accordingly present in so many quadrants of the framework), may explain why it has remained so popular over such an extended period of time, with no clear reduction in sight in the future either. By being used for knowledge sharing in different ways, it helps the frontend developers strengthen their mutual ties (Cross et al. 2001) and cognitive trust in others' knowledge (Choi et al. 2010). By affording overhearing (Gutwin et al. 2004), even the novices will quickly learn who the experts are in the company, and who can be asked for help in different matters. Aided by the weekly Friday meetings this facilitates in creation of shared understanding of locations of expertise of unevenly distributed knowledge network (Choi et al. 2010; Wegner 1987). Furthermore, online chat offers everyone a low-effort tool for presenting one's questions.

Berends et al.'s (2006) and Boh's (2007) frameworks present mutually complementary pictures about Skype chat use at Delta. In the former framework, Skype chat represents only 2 out of 24 knowledge sharing mechanisms possible. In contrast to this, in the latter case, Skype covers 3 out of 4 of the mechanisms. The reason for different levels of coverage can be explained with the conceptual difference in what the frameworks are modeling. While Berends et al.'s framework provides a terminology for different origination mechanisms of knowledge depending on the stakeholders (sharing person, receiver, management) and the types of problems (shared, personal, problem-neutral), Boh's (2007) framework addresses the modes in which sharing of different types of knowledge (codified or personalized) can take place informally or formally in an organization.

Therefore, we can conclude that Skype chat – as such a very simple communication tool – was used in a versatile manner at Delta to support problem-neutral diffusion and problem-related information retrieval in several different manners, covering both informal and formal, as well as personalized and codified modes as depicted by Boh's (2007) framework. Looking at the characteristics of frontend development portrayed in the beginning of the paper – ephemeral knowledge, essentiality of external knowledge sources, and the benefit of knowing the experts in each domain – as well as the ill-structured problem solving that permeates all software programming activity, it is natural that Skype chat, which can support knowledge sharing in a versatile manner, is used actively among software developers. This way they can tap into each other's knowledge sources and cope with the uncertainties intertwined in their work.

Discussion and Implications

The results of the study can be summarized as follows. In project-based knowledge work in which much of the relevant knowledge is ephemeral due to the fast-paced technological development, a synchronous CMC tool can be a powerful knowledge sharing tool. In our case company, the Skype chat tool was used overwhelmingly for a whole range of knowledge-sharing modes, both in informal person-to-person interactions and in a more formal collective trouble-shooting. Our empirical results indicate that in addition to supporting personalization strategy, even a simple communication tool, such as Skype chat, can be used – at least to some extent – in codification of knowledge. In terms of communication functions, Skype chat was used especially for *informing* (e.g., notifications about novel information) and for *peer help*. These purposes add to the previously reported uses that have been observed in project-internal collaboration and problem solving, such as communication about availability, technical work through chat, awareness, and social talk (Gutwin et al. 2004; Handel and Herbsleb 2002; Isaacs et al. 2002; Nardi et al. 2000a).

Previous research has raised a concern that instant messaging can be detrimental to work productivity due to the interruptions that it may introduce to work (Mansi and Levy 2013; Rennecker and Godwin 2003). Despite this, in our study the experts proved nevertheless willing to devote their time into helping each other. A positive attitude towards helping, especially when the knowledge flows from the experts to other less knowledgeable people in the organization, may have an important larger-scale effect on Delta's efficiency even if it may introduce inefficiencies on a personal level. This remains however to be shown in future research, as our study did not include a measurement of work performance neither on personal nor on company level.

Limitations of the Study

The main limitation of our study is that the results have been almost solely based on chat contents without any extraneous data. Being able to triangulate or contrast the findings with other data would have increased the external validity of our findings. Similarly, we could not use chat data metrics as independent variables in predictions of a dependent variable. A suitable dependent variable for such analyses would have been the amount of knowledge actually put into use in programming-related activities. However, quantifying the amount of knowledge is notoriously difficult and will require a dedicated undertaking that we are only now able to start, after the positive findings of our present analyses.

Another limitation is the completeness of our data. We segmented and classified only 6 out of 17 months of our data. This leaves an unanswered question whether our sample is fully representative of the overall activity of Delta's frontend developers. This limitation will be addressed in our future work. We will then also address the weakness arising from single-person coding that is vulnerable to possible coding errors.

Implications for Research

Our study holds several theoretical implications for knowledge sharing research. First, it demonstrates that attempting to postulate one ICT tool to represent one sharing mechanism only along any dimensions can be problematic. Our study presents an example in which a single ICT tool was actually used in a highly versatile manner, supporting multiple modes of knowledge sharing in Boh's (2007) framework and two origination mechanisms in that of Berends et al. (2006). Skype chat supported especially two communication functions between programmers: notifying about novel information and offering peer help. Previous research on chat use in organizations has identified several other purposes of use, including availability-related communication, cooperation in technical work, awareness, and social talk. It is likely that more communication functions for Skype chat can be identified in future studies.

Project-based knowledge work takes place in a fast-changing context where experts in a domain may be separated by allocation into different projects, as appears to be the case with frontend software development. This kind of environment may increase the extension of each ICT tool beyond their original, intended use, also to support different knowledge sharing mechanisms. The fact that technologies are used heterogeneously and creatively (Salovaara et al. 2011 2013) calls for carefulness if one wants to find connections between ICT features and their performance implications in a knowledge work context. In our case, even a very simple technology such as a text-based chat tool proved capable of contributing to work through a number of different knowledge sharing mechanisms.

Secondly, using two different frameworks to analyze the same data was revealing also in another way. Based on one framework (Boh 2007), Skype chat could not be mapped onto one position or axis in its dimensions while in the other framework (Berends et al. 2006), the full range of its knowledge sharing capabilities was obscured by the indigenous practices by which it had been adopted in the organization. For example, the Skype chat tool that we studied was not used for project-internal discussions or management-driven knowledge sharing, as commonly found in many software development organizations (e.g. Nardi et al. 2000a). Therefore, even after utilizing two frameworks, the competitive strengths possessed by Skype chat (and synchronous CMC more generally) remain ambiguous. The activeness in its use at Delta, nevertheless, begs for an answer.

In the present case, we believe that Skype chat's unique and competitive strength lies in its support for *managing ephemeral knowledge* in frontend development domain. Practically all knowledge sharing in the Skype chat channel addressed issues that are in a constant change. For instance, questions related to choices of a programming library or an interface widget, their updates to new versions, or uses of certain function calls are knowledge that will with high likelihood become irrelevant within a couple of years and be replaced with questions on new libraries, widgets, or function calls.

Therefore, we propose that more attention in knowledge management research should be given to ephemerality of knowledge and more generally on the fluctuation of relevance that knowledge may assume. The existing literature on knowledge management has hardly addressed this aspect at all. Based on our empirical study, we would suggest that synchronous CMC tools would be suitable for ICT-enabled sharing of ephemeral knowledge. Other media, such as documents shared through email, would suit for sharing more permanent or enduring knowledge. A practical as well as a theoretical challenge related to the relevance of knowledge is the fact that any piece of knowledge can, in principle, alternate between relevant and outdated states. The concept of "half-life" (cf. Siemieniuch and Sinclair 1999) may therefore not be a valid conceptualization due to its assumption on a monotonous decay of relevance of knowledge. This has implications for practical knowledge management: a system intended for sharing ephemeral knowledge should provide a capability to "resurrect" such pieces of knowledge that already are considered as outdated or irrelevant. Having such a capability would prevent organizations from "re-inventing the wheel" when facing the same needs for knowledge anew at a later time. Also the retrieval of past ephemeral knowledge poses challenges. If a piece of knowledge is considered ephemeral, workers will invest little effort in its indexing, tagging, and storing. Such actions must be automatized in order to make retrieval possible in the first place. The concept of ephemerality therefore poses both theoretical and practical questions for future research.

These two implications – a need to acknowledge ICT's versatility for heterogeneous uses in knowledge sharing and ephemerality as a crucial aspect in knowledge sharing – offer a number of possibilities for future research. Paying attention to these implications may improve knowledge management in domains characterized with high need for rapid knowledge exchange, importance of peer help in deliberation of one's decisions, and collaboration between experts across projects, as exemplified by frontend developers in software development in our study.

Acknowledgments

The authors want to thank Johanna Bragge for helpful comments. This work was supported by Academy of Finland through funding for *Creative use of ICT* project.

References

Alavi, M. and Leidner, D. E. 2001. "Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly* (25:1), pp. 107–136.

Berends, H., van der Bij, H., Debackere, K., and Weggeman, M. 2006. "Knowledge Sharing Mechanisms in Industrial Research," *R&D Management* (36:1), pp. 85–95.

Bhatt, G. D. 2011. "Knowledge Management in Organizations: Examining the Interaction Between Technologies, Techniques, and People," *Journal of Knowledge Management* (5:1), pp. 68–75.

Boh, W. F. 2007. "Mechanisms for Sharing Knowledge in Project-Based Organizations," *Information and Organization* (17:1), pp. 27–58.

Boh, W. F. and Wong, S. S. 2013. "Organizational Climate and Perceived Manager Effectiveness: Influencing Perceived Usefulness of Knowledge Sharing Mechanisms," *Journal of the Association for Information Systems* (14:3), pp. 122–152.

Choi, S. Y., Lee, H., and Yoo, Y. 2010. "The Impact of Information Technology and Transactive Memory Systems on Knowledge Sharing, Application, and Team Performance: A Field Study," *MIS Quarterly* (34:4), pp. 855–870.

Cross, R., Parker, A., Prusak, L., and Borgatti, S. P. 2001. "Knowing What We Know: Supporting Knowledge Creation and Sharing in Social Networks," *Organizational Dynamics* (30:2), pp. 100–120.

Daft, R. L., Lengel, R. H., and Trevino, L. K. 1987. "Message Equivocality, Media Selection, and Manager Performance: Implications for Information Systems," *MIS Quarterly* (11:3), pp. 355–366.

Davenport, T. H. and Prusak, L. 1998. *Working Knowledge: How Organizations Manage What They Know*. Cambridge, MA: Harvard Business School Press.

Faraj, S. and Sproull, L. 2000. "Coordinating Expertise in Software Development Teams," *Management Science* (46:12), pp. 1554–1568.

Grudin, J. 2006. "Enterprise Knowledge Management and Emerging Technologies," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS 2006),* New York, NY: IEEE Computer Society.

Gutwin, C., Penner, R., and Schneider, K. 2004. "Group Awareness in Distributed Software Development," in *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW 2004),* J. Herbsleb and G. Olson (eds.), New York, NY: ACM Press, pp. 72–81.

Hackbarth, G. and Grover, V. 1999. "The Knowledge Repository: Organizational Memory Information Systems," *Information Systems Management* (16:3), pp. 21–30.

Hakkarainen, K., Palonen, T., Paavola, S., and Lehtinen, E. 2004. *Communities of Networked Expertise: Professional and Educational Perspectives*. Amsterdam: Elsevier.

Handel, M. and Herbsleb, J. D. 2002. "What Is Chat Doing in the Workplace?," in *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW 2002),* E. F. Churchill, J. McCarthy, C. Neuwirth, and T. Rodden (eds.), New York, NY: ACM Press, pp. 1–10.

Hansen, M. T., Nohria, N., and Tierney, T. 1999. "What's Your Strategy for Managing Knowledge?," *Harvard Business Review* (77:2), pp. 106–116.

Hendriks, P. 1999. "Why Share Knowledge? The Influence of ICT on the Motivation for Knowledge Sharing," *Knowledge and Process Management* (6:2), pp. 91–100.

Hobday, M. 2000. "Project-Based Organisation: An Ideal Form for Managing Complex Products and Systems?," *Research Policy* (29:7–8), pp. 871–893.

Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D. J., and Kamm, C. 2002. "The Character, Functions, and Styles of Instant Messaging in the Workplace," in *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW 2002),* E. F. Churchill, J. McCarthy, C. Neuwirth, and T. Rodden (eds.), New York, NY: ACM Press, pp. 11–20.

Ko, A. J., DeLine, R., and Venolia, G. 2007. "Information Needs in Collocated Software Development Teams," in *Proceedings of the 29th International Conference on Software Engineering (ICSE 2007),* New York, NY: ACM Press, pp. 344–353.

Kuk, G. 2006. "Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List," *Management Science* (52:7), pp. 1031–1042.

Lee, A. S. 1994. "Electronic Mail as a Medium for Rich Communication: An Empirical Investigation Using Hermeneutic Interpretation," *MIS Quarterly* (18:2), pp. 143–157.

Leseure, M. J. and Brookes, N. J. 2004. "Knowledge Management Benchmarks for Project Management," *Journal of Knowledge Management* (8:1), pp. 103–116.

Linde, C. 2001. “Narrative and Social Tacit Knowledge,” *Journal of Knowledge Management* (5:2), pp. 160–170.

Lou, H., Chau, P. Y. K., and Li, D. 2005. “Understanding Individual Adoption of Instant Messaging: An Empirical Investigation,” *Journal of the Association for Information Systems* (6:4), pp. 102–129.

Mansi, G. and Levy, Y. 2013. “Do Instant Messaging Interruptions Help or Hinder Knowledge Workers' Task Performance,” *International Journal of Information Management* (33:3), pp. 591–596.

Markus, L. 2001. “Toward A Theory of Knowledge Reuse: Types of Knowledge Reuse Situations and Factors in Reuse Success,” *Journal of Management Information Systems* (18:1), pp. 57–93.

Nakakoji, K., Ye, Y., and Yamamoto, Y. 2010. “Supporting Expertise Communication in Developer-Centered Collaborative Software Development Environments,” in *Collaborative Software Engineering,* A. Finkelstein, J. Grundy, A. van den Hoek, I. Mistrik, and J. Whitehead (eds.), Berlin Heidelberg: Springer-Verlag, pp. 152–169.

Nardi, B. A., Whittaker, S., and Bradner, E. 2000a. “Interaction and Outeraction: Instant Messaging in Action,” in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW 2000),* W. A. Kellogg and S. Whittaker (eds.), New York, NY: ACM Press, pp. 79–88.

Nardi, B. A., Whittaker, S., and Schwarz, H. 2000b. “It's not What You Know, It's Who You Know: Work in the Information Age,” *First Monday* (5:1).

Niinimäki, T. 2011. “Face-to-face, Email and Instant Messaging in Distributed Agile Software Development Project,” in *Proceedings of the 2011 Sixth IEEE International Conference on Global Software Engineering Workshop (ICGSEW 2011),* New York, NY: IEEE Computer Society, pp. 78–84.

Nonaka, I. and Takeuchi, H. 1995. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* New York, NY: Oxford University Press.

Pazos, P., Chung, J. M., and Micari, M. 2013. “Instant Messaging as a Task-Support Tool in Information Technology Organizations,” *Journal of Business Communication* (50:1), pp. 68–86.

Rasters, G., Vissers, G., and Dankbaar, B. 2002. “An Inside Look – Rich Communication Through Lean Media in a Virtual Research Team,” *Small Group Research* (33:6), pp. 718–754.

Rennecker, J. and Godwin, L. 2003. “Theorizing the Unintended Consequences of Instant Messaging for Worker Productivity,” in *Sprouts: Working Papers on Information Systems* (3:14).

Robillard, P. N. 1999. “The Role of Knowledge in Software Development,” *Communications of the ACM* (42:1), pp. 87–92.

Ruuska, I. and Vartiainen, M. 2005. “Characteristics of Knowledge Sharing Communities in Project Organizations,” *International Journal of Project Management* (23:5), pp. 374–379.

Salovaara, A., Helfenstein, S., and Oulasvirta, A. 2011. “Everyday Appropriations of Information Technology: A Study of Creative Uses of Digital Cameras,” *Journal of the American Society for Information Science and Technology* (62:12), pp. 2347–2363.

Salovaara, A., Öörni, A., and Sokura, B. 2013. “Heterogeneous Use for Multiple Purposes: A Point of Concern to IS Use Models’ Validity,” in *Proceedings of the Thirty Fourth International Conference on Information Systems (ICIS 2013),* F. Pennarola, J. Becker, R. Baskerville, and M. Chau (eds.), Milan, Italy.

Serenko, A., Bontis, N., and Hardie, T. 2007. “Organizational Size and Knowledge Flow: A Proposed Theoretical Link,” *Journal of Intellectual Capital* (8:4), pp. 610–627.

Siau, K., Tan, X., and Sheng, H. 2007. “Important Characteristics of Software Development Team Members: An Empirical Investigation Using Repertory Grid,” *Information Systems Journal* (20:6), pp. 563–580.

Siemieniuch, C. E. and Sinclair, M. A. 1999. “Organizational Aspects of Knowledge Lifecycle Management in Manufacturing,” *International Journal of Human–Computer Studies* (51:3), pp. 517–547.

Simon, H. A. 1973. “The Structure of Ill-Structured Problems,” *Artificial Intelligence* (4:3–4), pp. 181–201.

Storey, C. and Kahn, K. B. 2010. “The Role of Knowledge Management Strategies and Task Knowledge in Stimulating Service Innovation,” *Journal of Service Research* (13:4), pp. 397–410.

Tampoe, M. 1996. “Motivating Knowledge Workers – The Challenge for the 1990s,” in *Knowledge Management and Organizational Design,* P. S. Myers (ed.), Boston, MA: Butterworth-Heinemann, pp. 179–190.

Turner, T., Qvarfordt, P., Biehl, J. T., Golovchinsky, G., and Back, M. 2010. “Exploring the Workplace Communication Ecology,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing (CHI 2010),* E. D. Mynatt, G. Fitzpatrick, S. Hudson, K. Edwards, and T. Rodden (eds.), New York, NY: ACM Press, pp. 841–850.

Van Maanen, J. and Schein, E. H. 1979. "Toward a Theory of Organizational Socialization," in *Research in Organizational Behavior,* B. M. Staw (ed.), Greenwich, CT: JAI Press, pp. 209–264.

Wegner, D. 1987. "Transactive Memory: A Contemporary Analysis of the Group Mind," in *Theories of Group Behavior,* B. Mullen and G. R. Goethals (eds.), New York, NY: Springer-Verlag, pp. 185–208.

Wenger, E. 1998. *Communities of Practice*. Cambridge, UK: Cambridge University Press.

Yu, T.-K., Lu, L.-C., and Liu, T.-F. 2010. "Exploring Factors that Influence Knowledge Sharing Behavior via Weblogs," *Computers in Human Behavior* (26:1), pp. 32–41.