MEDIATED SHARING AS SOFTWARE DEVELOPERS' STRATEGY TO MANAGE EPHEMERAL KNOWLEDGE

Complete Research

Antti Salovaara, Aalto University School of Business, Finland, antti.salovaara@aalto.fi

Virpi Kristiina Tuunainen, Aalto University School of Business, Helsinki, Finland, virpi.tuunainen@aalto.fi

Abstract

According to some estimates, half of the knowledge in software programming goes out of date every three years. This ephemeral nature of programming-related knowledge demands knowledge management practices that the two traditional strategies—knowledge codification and knowledge personalization—are not able to satisfactorily solve. In this paper, we analyse what ephemeral knowledge is and what requirements it has for knowledge management in the context of software programming. We present a case study on a software company whose developers' work is affected by the ephemerality of knowledge, and describe the practices that address the requirements created by ephemeral knowledge. In particular, we found that although individual developers' knowledge management practices were highly heterogeneous, they were a basis for a very efficient information monitoring, filtering and discussion system on a collective level. The primary technologies in this system—microblogging and shared instant messaging chat—were used in a manner that could be categorized neither as part of codification nor personalization strategy. Instead, they suggest a third knowledge management strategy that we label as "mediated sharing". We describe its operation with three characteristics. This provides a starting point for further research on how ephemeral knowledge could be managed.

Keywords: ephemeral knowledge, knowledge management strategy, mediated sharing, software development.

1 Introduction

Adapting to and acting upon the changes in the surrounding environment is one of the key criteria for business competitiveness. Taking advantage of changes in the business environment requires capability to absorb new knowledge and update previously held views. The idea of new knowledge surpassing older one in relevance implies a concept of *ephemerality,* that is, fluctuation of relevance. Knowledge may become out-dated over time and needs to be discarded, but it may also regain its relevance later. In the literature, the trend of decreasing relevance is often described in terms of "half-life of knowledge", defined as the amount of time within which knowledge erodes in a particular domain (Machlup, 1962). Studies have suggested that half-life varies across domains, being 3 to 5 years in software programming, for example (New York Times 1991; Kruchten 2008; Charette, 2013).

As ephemerality has quite clearly been recognized in knowledge-intensive working life—such as in software engineering—one would expect that knowledge management (KM) literature would have often discussed topics such as re-education of personnel or on-going maintenance of codified knowledge. However, to our best knowledge, topics such as these are mostly absent in the KM literature. Alavi and Leidner's (2001) well-known review of KM research mentions only that knowledge codification may result in unwanted rigidity and may compromise performance (p. 112).

Others have described ephemerality merely as a characteristic of project-specific knowledge (Leseure and Brookes, 2004).

To fill this gap in the extant KM literature, in this paper, we will present an empirical case study on knowledge sharing practices among software developers who must continuously update their knowledge of software libraries, computer platforms, and programming tools in order to stay up to date in technological progress. We will seek answers to two research questions: 1) *whether and how the ephemerality of knowledge can be observed in software developers' work*; and 2) *how the developers' knowledge management practices address the challenges of ephemerality of knowledge.* Based on our analysis, we will evaluate whether the existing KM strategies—codification and personalization (Hansen et al., 1999) in particular—can respond to the challenges of ephemerality. The findings suggest a need to consider a third knowledge management strategy that we call mediated sharing. This strategy entails knowledge workers' use of lightweight knowledge sharing tools that support personal knowledge management by rapidly distributing pieces of knowledge within and across organizational boundaries. By presenting the case study and its implications to knowledge management strategy literature, the paper will generate novel understanding of knowledge management in domains challenged by continuous change.

2 Defining Ephemeral Knowledge

Earlier research has presented numerous dimensions of knowledge, including for instance, tacit vs explicit knowledge (Polanyi, 1967; Nonaka and Takeuchi, 1995); internal (i.e., in-house) vs external (Cassiman and Veugelers, 2006; Menon and Pfefers, 2003); individual vs collective (Nonaka and Takeuchi, 1995); and the continuum from foreground to background knowledge (Bhatt, 2001). These dimensions, however, do not address the temporal dimension of knowledge, such as fluctuations of relevance and needs for continuous updating of one's knowledge.

Also the possibility of knowledge being or becoming out-dated has been acknowledged in earlier research (see e.g., Bhatt, 2001). However, a dedicated discussion on ephemerality of knowledge or the temporal fluctuation in its relevance has been scarce in KM literature. We build our definition of ephemeral knowledge on earlier KM literature. In the context of a study on management of project-based knowledge, Leseure and Brookes (2004) define ephemeral knowledge as knowledge that "is useful for one project but has a low probability of ever being used again" (Leseure and Brookes, 2004, p. 107). Its opposite is *kernel* knowledge, defined as "knowledge that need to remain and be nurtured within a company in order to sustain high project performance in the long-term" (ibid., p. 107). The wider question of whether knowledge could be ephemeral also beyond the project-specific context and what that would imply for knowledge management more generally is, however, ignored.

Siemieniuch and Sinclair (1999) focus more closely on the concept of half-life of knowledge and presentation of a lifecycle perspective to knowledge management. They observe that "there is the deterioration of value in knowledge because of the changing competitive environment", (p. 521). Their discussion on ephemerality focuses on the ways in which out-dated information can be recognized, but leaves open the questions on the preferred ways of acquiring new knowledge.

Motivated by the analyses of these few extant studies, we find that a more comprehensive conceptualization of ephemeral knowledge is needed. Our purpose is to build on the work of Leseure and Brookes (2004) and Siemieniuch and Sinclair (1999) and substantiate it with empirical material. We consider any piece of knowledge—whether project-based or something else—as having some level of ephemerality. While Leseure and Brookes (2004) postulated a dimension reaching from non-project-specific kernel knowledge to project-specific ephemeral knowledge, we consider a more general temporal continuum in which *stable* and *ephemeral* types of knowledge can be found in the opposite extremes. Stable knowledge is closely related to the concept of kernel knowledge (Leseure and Brookes, 2004): it remains true over extended periods of time and across different contexts. Ephemeral knowledge, in contrast, may remain empirically veridical (i.e., correspond to the observed reality) only for a relatively short period of time.

More specifically, stable knowledge consists of information whose foundations has not significantly changed over time and are believed to remain the same also in the future. For example, in software programming, design patterns that codify the best ways to program certain often-occurring software elements (Gamma et al., 1994) could be described as stable kernel knowledge. Even though programming languages, software platforms and programming tools have changed over time, the foundation on which design patterns have been built have remained largely unchanged, rendering design patterns a stable piece of knowledge.

Ephemeral knowledge, in contrast, consists of information that may be more recent or which are on a brink of becoming obsolete any time. For instance, knowledge about competing software libraries and standards are considered ephemeral: while software can be built based on any of the available libraries, it is done with an awareness that the global programmer community may abandon the adopted library, leading to its slower and slower maintenance and ultimately to its untrustworthiness.

Whether a piece of knowledge is stable or ephemeral depends on the shared but not necessarily explicitly discussed opinion within the relevant community of experts. Whether a piece of knowledge is stable or ephemeral is therefore based on a subjective perception that may also fluctuate over time. Because of this, a piece of knowledge may be considered originally as being ephemeral, and later as stable when more experience of its validity has been gathered. Subsequent events may turn it ephemeral again, if they point out errors in it, or if more useful knowledge emerges. This fluctuation resembles the processes of accumulation of scientific knowledge: novel theories are regarded as tentative (i.e., ephemeral) until more evidence accumulates verifying their value and making them a stable part of the shared scientific knowledge. Later, they may lose their relevance when replaced with new, competing theories (cf. Kuhn, 1962).

To summarize, the concept of ephemeral knowledge provides a temporal viewpoint to knowledge, which is independent of the extant knowledge dichotomies (see the beginning of this section for examples of dichotomies). Concerning the widely used tacit vs. explicit dimension, for example, we find that explicit knowledge can be both ephemeral (e.g., design sketches in project work; Leseure and Brookes, 2004) or stable (e.g., textbooks on physics). Tacit knowledge, in turn, is often deeply internalized knowledge and therefore most often stable, but may become ephemeral when the context changes and the old skills are not useful anymore. The tacit vs. explicit dimension does not concern the temporality of the knowledge, but the way in which it is transferable between the actors.

3 KM Strategies and Ephemeral Knowledge

KM literature has traditionally declared a distinction between codification and personalization strategies (Hansen et al., 1999; Boh, 2007). In the following, we will conceptually evaluate whether two strategies' usefulness in work contexts where a significant part of knowledge may be ephemeral.

The first of the two primary KM strategies, *codification,* refers to careful externalization of knowledge from experts and its storage in databases where it can be accessed and used easily by anyone in the organization (Hansen et al., 1999). A prime example is a knowledge repository that can be queried for information (Zack, 1999). In this strategy, context-specific details about the knowledge are removed (or retained in an easily understandable story-like format (Davenport and Prusak, 1998; Linde, 2001). Such a decontextualization may be problematic in work contexts dealing with ephemeral knowledge. Firstly, both updating of the knowledge and retrieving it require effort and employees easily forget them (see, e.g., Stenmark and Lindgren, 2004). Secondly, because the knowledge is weakly connected to the reality where it is intended to be used, it can easily transpire that a piece of knowledge remains unattended in the knowledge repository while the environment for which it has been applicable changes (Siemieniuch and Sinclair, 1999), or the codification cannot keep up with emergent knowledge processes (cf. Markus et al., 2002). As a result, codification strategy cannot cope with the requirements of managing ephemeral knowledge. We use these two prime problems to suggest requirements for a strategy suitable for contexts where ephemeral knowledge is prevalent:

Requirement 1: *Effortless content flow to relevant people*: Organizations need to be able to identify the knowledge that needs to be discarded or updated with newer content. This should not incur additional burden for the employees or else motivating them for this becomes difficult.

Requirement 2: *Keeping the knowledge connected to the context of use*: A piece of knowledge may become ephemeral because the context within which it is intended to be used changes. Organisations need to know when their knowledge contents need updating. This happens best if the knowledge is closely connected to the context where it is used.

Personalization is the other of the two widely acknowledged KM strategies, and has been recently more widely advocated (Earl, 2001; Ackerman et al., 2013). It is based on the idea of expertise sharing (Hansen et al. 1999). It entails creation of knowledge in teams and its transfer through collaboration and social interaction between people (e.g., Nonaka and Takeuchi, 1995). Several ways to implement this strategy have been developed, including creation of expert networks and being active members in them (e.g., Earl, 2001), establishing problem-solving teams (e.g., Nonaka and Takeuchi, 1995), and development of directories of experts and their skills with a purpose of facilitating finding experts (e.g., Whelan, 2013). In all of these implementations, personalization strategy meets the above-presented Requirement 2 excellently by bringing the organization in touch with people who have deep understanding and first-hand connection to its application. However, Requirement 1 is poorly met, because transfer of knowledge across the organization is slow if it is based on person-to-person interactions. As a result, while the personalization strategy works better than codification, it also has limitations in coping with ephemeral knowledge.

Another perspective to KM strategies is provided in Earl's (2001) classification of six "schools of strategy" that describe how companies manage their knowledge. Of the six schools, the systems and engineering schools are close to the codification strategy. The commercial school focuses on the business value of company's knowledge assets and is outside the scope of our study. Finally, cartographic, organizational, spatial and strategic schools aim at bringing people together and are therefore personalization-oriented. The Requirements 1 and 2 are compatible with the organizational school's aims, where the focus is on knowledge building in expert communities. Earl (2001) describes only in a general level the means by which successful expert communities can be nurtured. Our study can be seen as an attempt to concretise these aims.

While the dichotomy of codification and personalization strategies (Hansen et al.', 1999) can be used to depict a wide range of KM practices and approaches, including Earl's (2001) six schools of strategy, we find it insufficient for classifying some of the recently emerged knowledge sharing mechanisms facilitated by ICT tools, such as, enterprise social network services, instant messaging (IM) and micro-blogging (e.g., Yammer, Skype's chat functionality and Twitter, respectively). The use of these tools relies on person-to-person communication (i.e., personalization) but are based mostly on brief textual externalized communications (i.e., codification). These mechanisms have the advantage of raising employees' *awareness* of new knowledge as well as exposing them to a flow of information related to their peers' on-going work (Herbsleb et al., 2002; Gutwin, Penner and Schneider, 2004). If these tools are widely adopted in the organization, new knowledge can reach everyone rapidly and help people update their knowledge (Requirement 1). The conversation enabling nature of these tools allows the discussions to be tied with on-going work, and thereby the knowledge remains connected to the context in which the it is used (Requirement 2).

Based on this conceptual analysis, we conclude that these tools that cannot be fully comprehended with the dichotomy of codification vs. personalization strategies (Hansen et al.', 1999) may offer a solution to the needs arising from ephemeral knowledge. We will next present a case study of an organization where the employees were actively using such tools.

4 Case Study: Managing Ephemeral Knowledge in Frontend Development

This paper builds on our previous work (Salovaara and Tuunainen, 2013) on knowledge sharing in distributed software engineering teams and their instant messaging (IM) based knowledge sharing. The software company in question—Futurice (www.futurice.com)—has almost 200 employees and is focused on commissioned projects in which tailor-made software products (websites, web applications, mobile services etc.) are built for customers that come from different industries and are of different sizes. Our specific focus is on Futurice's 30–40 *frontend developers* who program the user interface related parts as well as the gateways from the different devices (computers, phones, tablets) to the backend servers.

In the present paper, we complement the conceptual analysis above by addressing our two research questions: whether and how ephemerality of knowledge can be observed, and how the developers' knowledge management practices address the challenges of ephemerality of knowledge. Our data is of three types:

1) *IM chat message corpus* from the frontend developers' single Skype discussion thread. The data covers a complete log of messages from a 28-month period (September 2011 to December 2013) and contains 25 945 messages.

2) *Interviews* on personal knowledge management and sharing practices. We interviewed six frontend developers over a course of five months. The long time span allowed us to interleave analysis and data collection and adapt the interview structure iteratively. This served well the exploratory theory-building oriented nature of our study. The interviews addressed informants' general perception of knowledge in software development as well as practices of knowledge acquisition, acting on it, and sharing it. Informants also filled in a "personal KM sheet"—a structured form that asked him to specify his sources of information, to list the ICT tools that he uses to obtain this information, to describe how he processes the information, and what ICT tools he uses in this process.

3) Counts of incoming and outgoing communication. We calculated how many programming-related messages one of our informants received and how many messages he sent during 25 days.

In addition, we participated seven times in frontend developers' weekly show-and-tell meetings that gave us better ability to interpret our data, and observed the informants demonstrating their use of different KM tools in practice during the interviews.

We carried out the interviews in the company premises, recorded them and transcribed afterwards. They lasted between 35 and 73 minutes. All the informants were male, between 28 and 37 years of age (average 31). They had been employed at Futurice from 1 month to 9 years (avg. 4), of which they had worked as frontend developers 1 month to 4 years (avg. 3). Before joining Futurice, they had worked as professional programmers between 2 and 6 years (avg. 4). Three informants had a M.Sc. degree and two had a B.Sc. degree from computer science or a related field. One had an MA degree in education.

In the analysis, we started by reading the interviews and making qualitative observations. After having developed an understanding of the personal KM practices, we started the analysis of the IM chat. By reading IM chat discussions, we collected a large set of technical terms such as library names. We then used Google Search to find out their competitors. For example, when searching for competitors to Backbone, we used "Backbone vs" as a query in Google to find blogs that compared frameworks. We quickly had lists of competing libraries and frameworks programming tasks, such as event handlers, CSS accelerators, dependency managers, and so on. We wrote a simple script in Python programming language that could search for keyword appearances in the IM chat and draw graphical visualizations of each term's appearance on a common timeline. Some of them are presented in the following section.

Finally, we read the interviews again and focused on those parts where the informants discussed ephemerality of knowledge, its temporal fluctuation of relevance, how they coped with it, and what

ICT tools they used. We analysed these parts (54 in total, some of which were ½ page long) and analysed them separately, searching for common categories. This process had elements of grounded theory method with respect to open and axial coding (Strauss and Corbin, 1990) but was not thematically entirely open-ended. Instead, the quotes were purposely sampled to contain only issues related to personal KM practices and nature of ephemeral knowledge (i.e., topics relevant to our research questions). As the very final step we quantified the amount of incoming and outgoing communication for one informant.

5 Findings

5.1 Evidence for ephemeral knowledge

Both the interviews and our analyses on the IM chat discussions confirmed that the ephemerality of knowledge can be observed in software developers' work. In interviews, the informants pointed out in particular that libraries and frameworks have the characteristic of being short-lived:

"Those UI [user interface] *libraries that have some new components, they keep on popping up all the time. Like, what is Twitter's Bootstrap, that's now really big, now there's Topcoat from Adobe, that's an exact competitor. And more responsive UI frameworks appear all the time. So that, what's the best at the moment is a moving target."* (Informant D)

"Ember.js, that has been around already for a while. But if I'm right, at that time [when it was released] we didn't have any new projects being launched here at Futurice, so we did not even have time to get started with Ember when Angular.js already came around. And now, Facebook has made React, which also tries to help frontend's most visible part, the view. And it's partly overlapping Angular.js. If you are using React, there's no point of using Angular." (Informant E)

With these two often interchangeably used terms—libraries and frameworks—the informants referred to open source software components that are used to accelerate JavaScript programming by providing structure, abstractions and higher-level commands for various crucial operations. The selection of the set of right frameworks for the project requires expertise. As the second quote above indicates, the libraries often have functional dependencies: programmers are not entirely free in their choice of frameworks, because choosing one will necessitate inclusion of another. Second, our informants told that there are usually several alternative software frameworks to choose from that differ in terms of two often conflicting characteristics of maturity and progressiveness. Third, we were told that the credibility of the framework developers' core team affects how developers perceive the quality of the library and whether they are willing to trust in the long-term commitment to the framework's development. Several factors are therefore at play when a project team makes decisions on the development libraries it will use.

In the frameworks' competition for popularity, the weaker frameworks lose their users and the detailed knowledge about these frameworks loses its relevance. Ephemerality of such knowledge could be observed also by analysing term frequencies in the IM chat. Figure 1 visualizes the decreasing frequency of discussion on Backbone and thereby its ephemeral relevance at Futurice. The terms following Backbone provide other examples. They are related to mobile phone manufacturer Nokia and its previous operating systems (i.e., Symbian, S40 and S60). Their frequency patterns reflect Nokia's and its operating systems' decreasing importance in the market and thereby ephemerality of knowledge related to them. The same applied to the knowledge of Windows Phone 7 (i.e., "wp7") that was replaced by Windows Phone 8 in 2012.

Interestingly, however, also iPhone and HTML5 portrayed decreasing trends although their relevance in the market was not decreasing. Several reasons may explain this. First, a decrease in frequency may indicate also stabilization of knowledge, in which case the technology or framework becomes so commonplace and familiar to everyone that it needs to be rarely discussed. Alternatively, other simultaneous processes might be at play, such as changes in the software projects that are in progress.

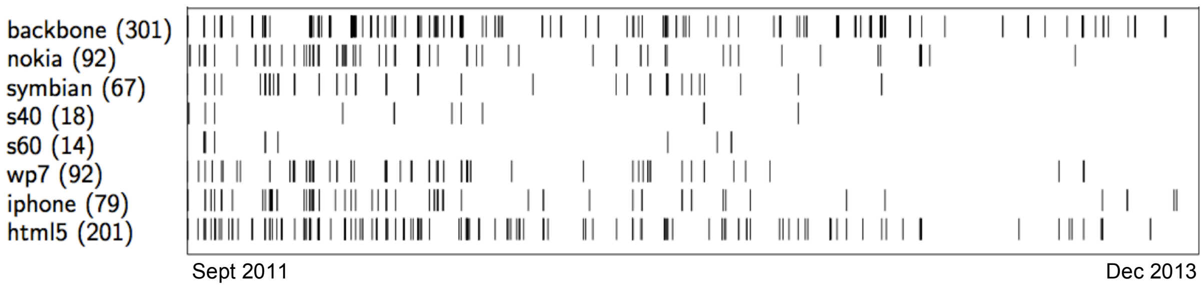

Figure 1. Frequency of term appearances in the IM chat for areas of knowledge that proved ephemeral.

5.2 Evidence for one area of knowledge losing its relevance to a competitor

In frontend development, many programming tasks can be accomplished in several alternative ways using different libraries and frameworks. The competition between the libraries sometimes leads to "revolutions" in which a previously dominant framework becomes replaced with another framework. The already mentioned migration from Backbone to Angular is a prominent example. Another revolution was the adoption of promise-based asynchronic programming such as client–server communication.

"A big insight during the last couple of years has been, we were quite early users in our project, the handling of asynchronic programming using promises instead of callbacks. That's now abstracted away into promises, and that increases the level of abstraction in the code and looks nicer. You can't live without them anymore, but first there was a lot of learning and puzzlement. Especially the first time when you had to start using them. During the last year the whole scene has had an enlightenment with them." (Informant B)

A third example was the widespread adoption of Grunt, a task runner for automating the steps in frontend software compilation. Command line based build scripts and a framework called Ant was in use, but no well-working solution for this task existed before its release:

"But one thing that has of course helped in many ways is that there is now a Node-based Grunt tool for automating build processes. Previously that work has been almost non-existent or some very custom-made handling [...] We had some custom scripts and Jake tool, which is a JavaScript version of Make. You could do some things with that, but that was a bit so and so [...] When you didn't Grunt which is accepted and liked by everyone, someone used Jake, another wrote Bash scripts, and a third used Python. Which meant that the result was quite a mess." (Informant B)

Figure 2 visualizes the three examples above—the adoption of Angular, programming with promises, and use of Grunt—and the frequency patterns in our IM chat data illustrating how new frameworks overtook the relevance from their competitors.

As Figure 2 shows, the discussion on Backbone did not die altogether after Angular's release. There is a "long tail" that overlaps the period when Angular started to interest the developers. It is a result of the legacy effect that sustains the relevance of popular frameworks. By having been the leading framework for at least two years, many projects continued to depend on Backbone and many developers were still using it. This, in turn, sustained discussions around it.

We conclude that ephemerality of knowledge can be observed in software developers' work, as demonstrated by the patterns for ephemeralized knowledge (Figure 1) and more complex patterns such as revolutions (Figure 2).

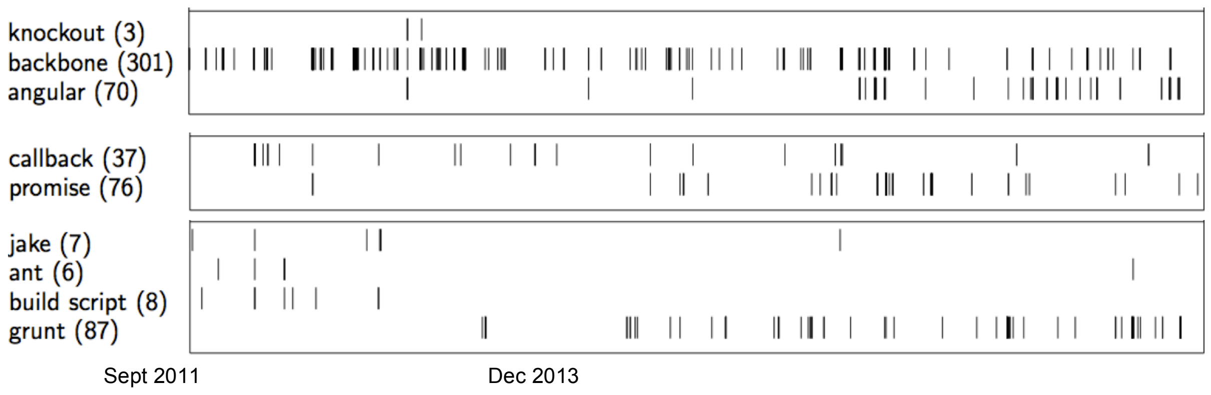

Figure 2. Three examples where one framework (the last one in each of the three boxes) became more actively discussed than its competitors, thereby overtaking its competitors' relevance.

5.3 How developers' KM practices address the ephemerality of knowledge

To attend to the question of how the developers' knowledge management practices address the challenges of ephemerality of knowledge, we analysed our informants' deliberations on the choice of frameworks (addressed in this sub-section) and the practices by which they kept themselves updated about the state of the art in frontend development (addressed in the two following sub-sections).

Although the quotes above showed that the frontend developers were aware that many of the frameworks will be only ephemeral and will not have a long lifetime, they waiting indefinitely to see which ones will survive would have compromised their productivity. Their challenge therefore was to select those frameworks that offered high utility and reliability but also had a high likelihood for a long lifespan.

We identified two approaches by which our informants strived for making the right decisions when choosing frameworks and libraries. The first approach for coping with the ephemerality of knowledge was *conservative adoption*. Although the developers were aware of several newer frameworks, they preferred tried and tested ones in their projects. In particular, developers tried to use framework combinations that had a wide global following and were therefore likely to remain actively developed also in the future. Developers tried to apply the same safe configurations across several projects also because familiarity speeds up development and decreases errors.

We named the other approach as *continuous staged learning*. As the next sub-section will show, the developers used several ICT tools to acquire information about the on-going progress in the field. In the most superficial stage, the learning consisted of scanning: following several news sources and browsing their contents through. Scanning did not need to be comprehensive or systematic. Because the news circulated in several ICT-based media, informants knew that they would eventually reach them:

"The biggest challenge is that there is crazy amount of information. About a year ago I followed those sources very carefully, hacker news and twitter, or reddit's certain sub-reddits, but […] *I think that I have now grown up a bit and learned to be a bit less neurotic. I have started to rely on myself. Anyway, I'm active in these things both at Futurice and outside* [in the community], *participated in local user group meetings. Finally the important things will emerge. If you participate in the meetings and scan the chat and weekly meetings, nothing big will go past your ears."* (Informant C)

"All in all there's so much incoming information, although I'm also following a very narrow field. But the things that you hear about are so much the same, it does not really matter if you miss 80% of it." (Informant D)

The next stage in continuous learning involved more engaged investigation. If a developer noticed news about a certain framework repeatedly, he invested more time for learning about it. The investigation involved visits to the framework's website and other sources in the Internet, and sometimes also small-scale programming, usually on the developer's own time. A considerable amount of time could pass between the first news-based exposure and the engaged investigation. Our informants told that postponing the engaged investigation about novel frameworks was intentional. It helped them become more confident that the time they put into investigation would pay off:

"Yes, I have all the time a list of things in my mind that I think I should know more about. But sooner or later you bounce into them anyway." (Informant B)

In the IM chat, we could observe different stages of continuous learning. The first remarks (see Table 1) were anecdotal, but gradually the developers started to express their specific interest about the framework and ask for more information. Finally, the questions became more specific, indicating that the framework had already been adopted.

1. First references:

2 March 2012 (the first time that Angular is mentioned)	A:	anyone got experience on knockout.js?
	B:	didn't C do some hacking with it at some point? I personally planned to try it long time ago, but never had the time. Another similar lib worth trying might be Angular.js
	C:	It seems that MVC-type of libraries (Spine, BB [Backbone], Angular, Ember? etc) are all solving the same problem. Thus, just pick a good tool and learn it well. No real need to switch between projects.
3 August 2012	D:	any of you tested JavascriptMVC as an alternative to Backbone or Spine?
	E:	AngularJS might be a valid alternative, haven't tried though

2. Explicit expressions of interest:

20 March 2013	F:	which are the current competitors for backbone.js, those that can be taken seriously? i guess still not that many around?
	G:	F: at least AngularJS is interesting. or well, I know smart people who take it quite seriously. maybe Ember and BatmanJS too? (though i've heard not-so-good things about using batman on a large project.)
3 April 2013	H:	seeing too much hype on angular that I think I cannot ignore it anymore ☺

3. Detailed questions and remarks:

25 July 2013	I:	Really cool thing I learned about AngularJS today: you can assign promises to scope and Angular replaces it with the value once the promise is resolved
1 August 2013	J:	I struggle with an AngularJS issue: scoping in directives
4 September 2013	K:	had a tremendous time setting up e2e tests for angular with karma today

Table 1. Stages of increasing interest in Angular in the IM chat.

5.4 Heterogeneity of ICT tools in personal knowledge management

We found that each programmer had developed his individual way of staying up to date about the on-going developments in the frontend development community. These different ways were in almost all of the cases ICT-based, although also face-to-face user meetings were mentioned. The number and diversity of different personal KM tools surprised us: 34 tools and channels belonging to 20 different categories, including both information sources (e.g. email newsletters, web forums, and RSS readers) and information storages (e.g., browser bookmarks, offline content readers, and podcasts). This list is a conservative estimate, since it is likely that all the sources of information were not mentioned.

Informants had personalized processes for handling the information flows from the source tools to storages. In the most common flow pattern, a piece of news was received through Twitter, a newsletter, RSS feed or a developer web forum. It contained a short synopsis of the news content accompanied with a link to a more extensive blog post, framework's website, or other article in the Internet. The developer opened the link and left it open in the web browser, where it could stay unattended for weeks or even months. At a later time, if the developer still found the content relevant, he stored it in an offline content storage (e.g., Instapaper) or an e-reader. At each stage, the developer filtered away those pieces of news that did not seem relevant enough for a more detailed attention. There was a lot of individual variation in the use of these tools and their combinations.

5.5 Collective filtering of knowledge across organizational boundaries

In addition to the individual-level heterogeneity in the choice of personal KM tools, our data suggests that the developers' individual efforts create a highly effective *collective knowledge filter.*

Figure 3 provides a simplified visualization of this collective filter's operation and presents some statistics on informant C's personal KM activity over the period of one month. While a comprehensive analysis of Futurice's frontend developers' knowledge sharing networks and patterns would be out of this paper's scope, we use a single informant to illustrate his use of heterogeneous ICT tools for the benefit of all Futurice's frontend developers. We limited our data collection to three knowledge sources: a web forum (Reddit's two discussion spaces called "sub-reddits"), the JavaScript Weekly newsletter and the tweets of all the Twitter contacts that C was following. Figure 3 shows the volume of communication in these channels—1760 tweets or re-tweets, 617 new discussions in Reddit (each containing several messages), and 24 news highlights—that constitute a theoretical upper bound of knowledge that C would have been exposed to if he had committed the time to read through all the communication in these channels.

While we know that the developers do not attend to all the content that they have received (see section 5.3.), we do not know exactly how much of the knowledge C attended to. We do see, however, from the records of outgoing communication, that during this particular month, C tweeted or re-tweeted only 13 messages, and contributed to Futurice's IM chat with 250 messages, 109 of which provided hyperlinks to content in the Internet. We believe that this one-person analysis of knowledge filtering provides indicative evidence that collective content filtering is both a necessary and a powerful

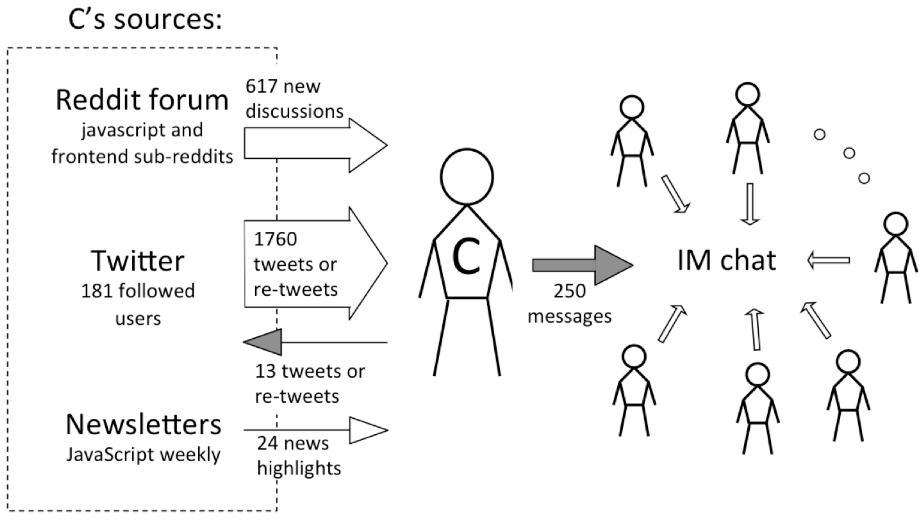

Figure 3. Example of a frontend developers' collective information filter. The values shown are the numbers of messages received (white arrows) and sent (gray arrows) by informant C between 1–25 November 2014 in some of his communication channels.

method for frontend developers to manage the information overload in their personal knowledge management.

More conceptually speaking, we suggest that the collective filter emerges from the joint activity where all the developers exhibit similar knowledge sharing behaviour. The efficiency of the filter is based on the following three characteristics. First, each developer is both a recipient and a source of information for others. Second, each developer filters knowledge by forwarding or creating less messages than he personally receives. Third, developers tend to specialise on certain topics in their information monitoring, sharing, and creation, thereby having a content-based division of labour. Most developers have special fields of interest within which they follow news more actively, or on which they work on themselves. Other developers with similar interests can use these higher-expertise peers as information proxies, relieving them from the effort of seeking the most relevant news in the same topic area. This is an effective way of decreasing noise and unnecessary content in the information channels that one follows.

In our case study, the most important tool in the filter was Twitter and its re-tweeting feature in particular. In the interviews, the informants told that it was often better to follow selected experts than to attempt to follow the tweets from each original source directly. The developers could rely on the fact that if a new piece of information was relevant enough, it reached them eventually anyway.

The other important filter tool complementing Twitter's functionality was the IM chat. While Twitter served as a source of external knowledge and information, the IM chat was used within the company for internal knowledge sharing, as shown in the quotes from our informants. In our previous analysis we found that IM was used especially for informing about new issues (31%), peer help (30%) and remarks on programming-related details (11%) (Salovaara and Tuunainen, 2013).

Although the personal KM methods were highly heterogeneous across individual developers, these two ICT tools—microblogging and IM chat—served as unifying knowledge sharing mechanisms that all the developers at Futurice used actively.

6 Discussion

In this paper, we examined ephemerality of knowledge and its implications for knowledge sharing practices among developers in a software company. We defined ephemeral knowledge as information that the focal community believes to become out-dated as the time passes because the context in which the knowledge is intended to be used is expected to change. Ephemerality of knowledge complicates decision-making because it decreases decisions' long-term trustworthiness. It is therefore a potentially vital challenge for decision-making and KM. In our study on frontend developers, a prime example of such a decision-making problem was the choice between competing software libraries.

We used interviews and term frequency visualizations to show that software development knowledge includes ephemeral knowledge and that ephemerality is a challenge that developers are constantly facing. We also showed that developers had varying individualized ways of coping with this challenge. Despite the heterogeneity of personal KM practices, on a collective level these practices complemented each other by producing a highly efficient emergent knowledge filter. This filter transcended organizational boundaries and linked the developers to the global developer network.

6.1 Limitations

Our exploratory case study unavoidably has limitations and leaves many issues in need for further investigation. First, it is not clear how well term frequencies can be used as proxy measures of knowledge relevance. In particular, interpreting what absence of discussion means is complicated. If something is not discussed, it may mean that the topic is not relevant or that it is generally well understood and therefore not in a need of discussion. Future research should develop new measures that would help disentangle the two interpretations and explain other ambiguities.

Another limitation is related to the difficulties in quantifying the volume of knowledge sharing. Our last analysis addressed only a limited set of channels of one informant. To study collective information filtering in more detail, a more comprehensive measure is needed. Finally, space did not allow us report individual personal KM practices in more length.

6.2 KM strategy based on mediated sharing

Based our conceptual analysis, we concluded that ephemeral knowledge poses two requirements for organizations' KM processes. First, ephemerality of knowledge presupposes mechanisms for effortless flow of content to the relevant people who can then replace the old with new knowledge (Requirement 1). Second, in order to identify when knowledge needs to be discarded and replaced, it must stay connected to its context of use and not be stored in a decontextualized manner in a detached knowledge repository where it can be forgotten (Requirement 2). We evaluated how the well-known KM strategies of codification and personalization (Hansen et al.'s, 1999) are able to tackle the challenges of ephemerality of knowledge. We noted that especially the recent lightweight messaging-based information sharing tools such as IM chats as well as social networking and microblogging services (e.g., Yammer and Twitter) are difficult to map to either of the two strategies. Moreover, our empirical case study suggested that it is these tools that are actually used in a context where ephemeral knowledge is a continuous challenge. In short, the lightweight messaging-based information sharing tools seem to suit for the developers' management of ephemeral knowledge and seem to be missing a place in the codification–personalization dichotomy (see Table 2).

As an implication of this, we propose a conceptualization of a third KM strategy that would complement the well-established codification and personalization strategies (Hansen et al., 1999). Table 2 illustrates the need for introducing the third strategy. While the mechanisms and tools in the codification strategy meet neither of the two requirements and the ones in the personalization strategy can answer only to the connectedness-related requirement, the previously uncategorized novel ICT based information sharing tools meet both of the requirements. Their similarities can be described with the following common characteristics:

- *Point-to-pointness:* Knowledge flows directly from producers to consumers without intervening storages. This makes mediated sharing stand apart from codification-based sharing where knowledge is stored in repositories before it is used. Point-to-pointness supports effortless content flow to relevant people (Requirement 1) by removing intervening storages.
- *Boundlessness:* Knowledge sharing may take place also between strangers and it can transcend organizational boundaries. For example, in our case study, frontend developers used Twitter to receive new knowledge from the global community. Boundlessness accelerates knowledge sharing by enabling unhindered flow of content (Requirement 1). Physical boundlessness makes mediated sharing differnet from personalization-based sharing, which usually depends on effortful rich communication that usually is possible only when the parties are in the same space.
- *Piecewiseness*: Knowledge is shared in small pieces that are effortless to produce and consume. This feature allows conversation-like knowledge sharing also when the sharing takes place through an ICT-mediated channel. In this way, piecewiseness helps to keep the shared content connected to the context (Requirement 2).

We argue that *mediated sharing* describes particularly well, with a single term, the three characteristics by which ephemeral knowledge can be addressed, and captures the essential characteristics of the ICT based tools that our inquiry found lacking a category. Similarly to the personalization strategy, also the mediated sharing strategy is based on communication. However, while personalization is about transfer of tacit knowledge through collaboration and socialization, the sharing mechanism in the mediated sharing strategy is lighter, by being mediated by an ICT-based communication medium.

Mechanisms and tools	**Requirement 1: Effortless content flow to relevant people**	**Requirement 2: Connectedness to the context of use**
Codification strategy:		
Company-wide digital knowledge repositories	No	No
Informal document exchange		
Personalization strategy:		
Creation of expert networks and being a member in them		Yes
Teamwork		Yes
Expert directories	No	Yes
Hallway conversations and informal social activities		Sometimes
Organized meetings and training sessions across organizational units		No
Mediate sharing strategy:		
Social network sites		
Instant messaging	Yes	Yes
Microblogging services		

Table 2. Knowledge sharing mechanisms in three strategies and their suitability for meeting the KM requirements of ephemeral knowledge.

In Earl's classification of KM strategies (2001), mediated sharing strategy shares characteristics of the technocratic engineering school on one hand, due to its focus on knowledge flows, and behavioural organizational school on the other, due to its focus on networks and communities. The resemblance with the engineering school is however mostly coincidental; although Earl mentions knowledge flows he considers them in a context of shared databases. Our approach, in contrast, is messaging-based. The organizational school is therefore most compatible with mediated sharing. One difference, however, is that mediated sharing is based on a decentralized point-to-point principle. Earl, in contrast, recommends that knowledge sharing is facilitated by a "human hub (or moderator)" (p. 225).

Future research should investigate whether the three characteristics are necessary and sufficient criteria for an ICT tool that should support sharing of ephemeral knowledge. None of the three tools listed under the new KM strategy in Table 2 possesses all of these characteristics, alone. However, IM and micro-blogging have these characteristics when their use is combined. Our case study showed how knowledge workers might orchestrate their use in a self-organized manner. Also a combination of micro-blogging and social networking sites would possess the characteristics together. This combination may form a valid solution for managing ephemeral knowledge in a different organization. Finally, we are not aware of any current single tool that would have all the five characteristics.

In this paper, we have presented the first more articulated definition for ephemeral knowledge, presented two requirements with which it challenges the existing knowledge sharing mechanisms and KM strategies, presented an empirical case study on the management of ephemeral knowledge, suggested five characteristics for ICT tools that may meet the two conceptually derived requirements, and finally proposed a new KM strategy of mediated sharing. More verification and conceptual work is required before the strategy of mediated sharing can be fully conceptualized and ultimately confirmed to be theoretically useful. This offers many opportunities and a new research direction for future Information Systems and KM research.

7 Acknowledgments

The authors wish to thank Academy of Finland for funding through the Creative use of ICT project and Aleksandre Asatiani for helpful comments.

References

Ackerman, M. S., J. Dachtera, V. Pipek and V. Wulf (2013). "Sharing knowledge and expertise: The CSCW view of knowledge management." *Computer Supported Cooperative Work* 22 (4–6), 531–573.

Alavi, M. and D. E. Leidner (2001). "Knowledge management and knowledge management systems: Conceptual foundations and research issues." *MIS Quarterly* 25 (1), 107–136.

Berends, H., H. van der Bij, K. Debackere and M. Weggeman (2006). "Knowledge sharing mechanisms in industrial research." *R&D Management* 36 (1), 85–95.

Bhatt, G. D. (2011). "Knowledge management in organizations: Examining the interaction between technologies, techniques, and people." *Journal of Knowledge Management* 5 (1), 68–75.

Boh, W. F. (2007). "Mechanisms for sharing knowledge in project-based organizations." *Information and Organization* 17 (1), 27–58.

Boh, W. F. and S. S. Wong (2013). "Organizational climate and perceived manager effectiveness: influencing perceived usefulness of knowledge sharing mechanisms." *Journal of the Association for Information Systems* 14 (3), 122–152.

Cassiman, B. and R. Veugelers (2006). "In search of complementarity in innovation strategy: Internal RandD and external knowledge acquisition." *Management Science* 52 (1), 68–82.

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

Earl, M. (2001). "Knowledge management strategies: Toward a taxonomy." *Journal of Management Information Systems* 18 (1), 215–233.

Gamma, E., R. Helm, R. Johnson and J. Vlissides (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley.

Gutwin, C., R. Penner and K. Schneider (2004). "Group awareness in distributed software development." In: *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW 2004)*. Ed. by J. Herbsleb and G. Olson. New York, NY: ACM Press, pp. 72–81.

Hansen, M. T., N. Nohria and T. Tierney (1999). "What's your strategy for managing knowledge?" *Harvard Business Review* 77 (2), 106–116.

Herbsleb, J. D., D. L. Atkins, D. G. Bayer, M. Handel and T. A. Finholt (2002). "Introducing instant messaging and chat in the workplace." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2002)*. Ed. by D. Wixon. New York, NY: ACM Press, pp. 171–178.

Kruchten, P. (2008). "The biological half-life of software engineering ideas." *IEEE Software* 25 (5), 10–11.

Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. Chicago, IL: University of Chicago Press.

Leseure, M. J. and N. J. Brookes (2004). "Knowledge management benchmarks for project management." *Journal of Knowledge Management* 8 (1), 103–116.

Linde, C. (2001). "Narrative and social tacit knowledge." *Journal of Knowledge Management* 5 (2), 160–170.

Markus, M. L., A. Majchrzak, and L. Gasser (2002). "A Design Theory for Systems that Support Emergent Knowledge Processes." *MIS Quarterly* 26 (3), 179–212.

Machlup, F. (1962). *The Production and Distribution of Knowledge in the United States*. Princeton, NJ: Princeton University Press.

Menon, T. and J. Pfeffer (2003). "Valuing internal vs. external knowledge: Explaining the preference for outsiders." *Management Science* 49 (4), 497–513.

New York Times (1991). "Engineer supply affects America."

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

Polanyi, M. (1967). *The Tacit Dimension*. London, UK: Routledge & K. Paul.

Salovaara, A. and V. K. Tuunainen (2013). "Software developers' online chat as an intra-firm mechanism for sharing ephemeral knowledge." In: *Proceedings of the Thirty Fourth International Conference on Information Systems (ICIS 2013).* Ed. By F. Pennarola, J. Becker, R. Baskerville, and M. Chau.

Siemieniuch, C. E. and M. A. Sinclair (1999). "Organizational aspects of knowledge lifecycle management in manufacturing." *International Journal of Human–Computer Studies* 51 (3), 517–547.

Stenmark, D. and R. Lindgren (2004). "Integrating Knowledge Management Systems with Everyday Work: Design Principles Leveraging User Practice." In: *Proceedings of the Hawaii International Conference on System Sciences (HICSS 2004).* New York, NY: IEEE Computer Society, pp. 1–9.

Strauss, A. L. and J. Corbin (1990). *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Thousand Oaks, CA: Sage Publications.

Whelan, E. and R. Teigland (2013). "Transactive memory systems as a collective filter for mitigating information overload in digitally enabled organizational groups." *Information and Organization* 23 (3), 177–197.

Zack, M. H. (1999). "Managing codified knowledge." *Sloan Management Review* 40 (4), 45–58.