Knowledge Creation and Loss within a Software Organization: An Exploratory Case Study

Davi Viana, Tayana Conte
USES Research Group
ICOMP/UFAM
Amazonas - Brazil
{davi.viana, tayana}@icomp.ufam.edu.br

Sabrina Marczak
Computer Science School
PUCRS
Rio Grande do Sul - Brazil
sabrina.marczak@pucrs.br

Raymundo Ferreira
INDT
Amazonas - Brazil
ext-raymundo.junior@microsoft.com

Cleidson de Souza
Instituto Tecnológico Vale and UFPA
Pará - Brazil
cleidson.desouza@acm.org

Abstract

Software development activities are very critical, since most of them heavily depend on individuals' knowledge and their capabilities. This knowledge must be retained and managed in order to avoid productivity breakdowns. This paper empirically explores how knowledge is created and lost in a software organization, and discusses implications for software development. This is done through a case study in which we investigated a Brazilian R&D Institute. We found that knowledge creation can be achieved through: (1) knowledge sources; (2) architectural frameworks that contain common applications and architectures for a set of projects; and (3) lessons learned that contains concerns regarding previous projects. Additionally, we verified that some teams' actions might influence knowledge loss. To identify and to understand which aspects are related to managing knowledge is the first step towards avoiding its loss and facilitating organizational learning. Our work contributes to this end.

1. Introduction

Software development is a knowledge intensive endeavor composed of several activities of socio-technical nature [1]. In this sense, carrying out such activities, which involve a high amount of knowledge, is not trivial [2]. Knowledge used and produced during these activities can be about technologies, software engineering methods, and/or the organization's internal processes [3]. Creating and disseminating knowledge within software organizations require commitment from practitioners and an organizational culture that favors such creation and dissemination activities [4].

Nonaka et al. [5] classify knowledge into two types: tacit and explicit. Explicit knowledge, also called codified knowledge, is considered to be transferable using a formal and semantic language. Moreover, this type of knowledge can be represented in documents and databases. On the other hand, tacit knowledge is based on the person's experience and is harder to formalize. Both types of knowledge are considered to be the main competitive assets of a software organization [1, 6]. This means that software organizations need to carefully manage knowledge [7].

The lack of knowledge dissemination within an organization can eventually cause the loss of important information. Mendonça et al. [2] states that loss of information can occur due to the following reasons: (1) team members who quit their job; (2) solutions that are forgotten; (3) lack of documentation of solutions, or when documented, they are not distributed in the right way; and (4) constant technological updates, causing severe issues for the organizations. Additionally, Mitchell and Seaman [8] corroborated that problems with project information storage and retrieval can cause obstacles in knowledge flow. These previous works suggest that it is important to analyze the organizational software development environment in order to create strategies to avoid knowledge loss.

In this paper we sought to empirically investigate how knowledge is created and lost within a software organization. Such analysis is important because software organizations want to retain knowledge created in their projects as much as possible to avoid rework and performance breakdowns. Our research question is: "*How is knowledge created and lost in a software organization?*" To answer our research question, we conducted an exploratory case study in a mid-sized software organization located in Brazil. Such organization is a non-profit Research and Development (R&D) Institute focused on the generation of new concepts, products and solutions for areas related to mobile technologies and the Internet. We conducted 28 on-site interviews and applied grounded theory techniques [9] to draw conclusions about our data.

Our results show that the software organization has a set of activities in place to support knowledge creation. The organization possesses relevant technical knowledge (software routines and components) that is

1530-1605/15 $31.00 © 2015 IEEE
DOI 10.1109/HICSS.2015.477

embedded into the source code of the developed software. When such source code is common and shared among different products, an architectural framework is created/updated to maintain the most updated knowledge about the developed products. However, not all knowledge can be created and maintained in that framework. In this sense, knowledge regarding the development process, and the decision-making regarding the adoption of technologies as well as the technical knowledge about the software products are lost due to the lack of adequate documentation of that knowledge. We identified indicators of the loss of some knowledge due to the poor management of certain software development activities.

In summary, our findings can support the understanding, in a deeper way, of the important issues regarding knowledge creation and loss in a software organization. This understanding can be used by management to organize the knowledge management (KM) practices in their teams and help them to identify how the organization as a whole can benefit from the knowledge generated at a project level. We also anticipate that the findings reported in this paper will support and guide future work conducted in this field.

The remainder of this paper is organized as follows. Section 2 presents related work to this research. Section 3 describes the research method and the study settings. Section 4 presents our qualitative analysis and Section 5 discusses our findings. Section 6 concludes the paper with final considerations and future work.

2. Related Work

Software team members often hold knowledge about the product being developed, the tools and techniques to support their work and the organizational processes [10]. However, there are situations in which a software developer might not have enough knowledge for performing his/her role, e.g., when the person is a newcomer to the team and has not learned about the product yet. Therefore, retaining and managing knowledge as well as making this knowledge available are an important aspect to support development activities in any software organization.

Activities regarding the creation and the dissemination of knowledge are part of the KM process [11]. Bjørnson and Dingsøyr [12] conducted a systematic literature review about the concepts that have been studied, the results and research methods that have been used regarding KM in software engineering. Besides presenting an analysis of the KM schools within the reviewed work, these authors conclude that the type of activities of KM to be applied in an organization will depend on how the software development is done. This is explained by the different implications of KM for agile and traditional development. Nevertheless, such review does not detail what activities help to *create* knowledge in software organizations.

Many researchers aimed at analyzing software process activities for creating knowledge and avoiding the loss of such knowledge. For instance, Aurum et al. [13] presented a case study on practices for KM in two Australian software organizations. The authors identified that the creation of knowledge was performed both implicitly and explicitly within the organizations. Their results showed that practitioners recognized that the knowledge creation was being performed in their software development projects. Another conclusion from this study was that the team meetings were acknowledged as crucial opportunities for practitioners to present new ideas, to offer advice, and to commit to processes and methodologies, creating new knowledge [13].

One type of team meetings is the project post-mortem meeting [14] (also namely review/retrospective meeting in agile methodologies context). During these meetings it is possible to gather and create new knowledge for the organization [14]. The postmortem is a collective activity that can be performed at the end of a project phase or at the end of the project. The motivation for performing a postmortem meeting is to know what happened in the project in order to improve future practices. The goal of the postmortem is to become a learning opportunity and not to evaluate the project. Similarly, Dingsøyr et al. [15] pointed out that the experiences created in this type of meeting assist the organizational learning because there is a discussion of past successes and past failures. Despite the fact that post-mortem meetings are important for knowledge creation [14, 15], it is necessary to verify during the software development process if there are other moments in which it is possible to create knowledge.

Mitchell and Seaman [8] use the KM technique of "knowledge mapping" as a research technique to characterize types of obstacles on knowledge flow in software projects. The characterization of such obstacles can assist the improvement of the software process. The researchers verified that software engineers strongly depend on explicit project document storage/retrieval and tacit internal team communication. Thus, it is important to deal with the obstacles regarding these dependencies, such as: the lack of a versioning mechanism and different needs in domain knowledge and technical knowledge. Additionally, Kukko and Helander [16] identified a set of barriers for K as a whole. Such barriers can negatively influence knowledge creation, causing the

loss of relevant information in the projects. Through a theoretical study, the authors described several barriers, including: lack of dissemination on the real benefits of knowledge sharing, lack of trust among the practitioners within an organization, low conscious of the value of the possessed knowledge, and lack of an adequate structure to handle and to maintain the knowledge. The identification of obstacles and barriers [8, 16] can support software organizations when applying certain strategies to avoid knowledge loss.

In general, previous results suggest that understanding certain activities for knowledge creation are important to support software activities and avoid possible knowledge loss. We will later come back to this related work to compare our findings.

3. Research Design

To understand the complex phenomenon of knowledge creation and loss in software organizations, we chose to conduct an exploratory case study [17]. This type of case study is carried out to verify what is going on in the real world and to generate insights and ideas [18]. The case must be an event from real life [17]. We chose an organization that is pioneer in the development of mobile software and that possesses other offices in Brazil. Additionally, this organization has an initiative to implement KM processes to support and to improve its software development process.

In our study, we used semi-structured interviews to collect data. We conducted 28 interviews with software practitioners who play different roles in the organization. Initially, we prepared a questionnaire with open questions about knowledge management, organizational learning, and the organizational environment. Table 1 presents a summary of a plan for our case study.

Table 1. Plan for case study based on [18].

Elements	Description
Objective	Exploratory
The case	The entire organization (four projects)
Theory	There is no specified theory [18]
Research questions	How is knowledge created and list in a software organization?
Methods	Direct (interviews)
Selection strategy	The software organization needs to retain knowledge within the organization itself

Then, we applied such questionnaire with software practitioners. After that, we transcribed and analyzed all interviews using grounded theory techniques [9]. We chose a qualitative method because such type of method supports a better comprehension of the issues that need a more specific and detailed analysis. Seaman [19] states that the use of qualitative methods allows the researcher to consider human behavior and thoroughly understand the object of study. By applying a qualitative method, we intended to obtain a more adequate understanding of the creation and loss of knowledge in software organizations. More details about the organization, data collection and analysis methods are presented below.

3.1. Study Context

The study took place in a software organization that is a research and development (R&D) institute that focuses on products and solutions for areas related to mobile technologies and the internet. Such organization is committed with the creation of technological solutions that generate value. Its focus on productization complements the R&D cycle, and accelerates technological development in Brazil. The organization has about 255 employees, distributed in different software projects. Such projects are related to mobile applications (apps), cloud services and operational system for mobile phones. The organization adopts several technologies, including C#, C++, J2ME, J2EE, Lua, and Hadoop. With the goal of maintaining knowledge always available for its practitioners to improve their productivity, the organization needs to better understand how knowledge is created within its context. The organization has recently defined a knowledge management initiative to better organize knowledge considered relevant to support the software development work performed by the software teams. This initiative aims to allow the practitioners to create and to improve their knowledge practices.

The organization has a well-defined hierarchical structure. The practitioners report to two types of managers: the project manager, who is responsible for supervising the projects; and the line manager, who is responsible for allocating people in projects and dealing with organizational and professional career matters. There is also a project leader role. This role is assigned according to the needs of the project manager. Normally, such need arises when a manager has to supervise too many projects and needs help with it. Developers are commonly allocated to only one project at a time. Designers and testers are shared resources across projects, i.e. they are assigned to more than a project at a time.

The software process is defined according to the project specifications and project manager needs. Such process is controlled by the team itself and is normally managed by the project manager when there is not a leader available. The teams aim at carrying out as many Scrum practices [20] as possible. Some team members hold the Scrum Master role. However, these

members cannot supervise all projects at a time given the high workload. Therefore, when the team does not officially allocates a Scrum Master, someone inside the team carries out the Scrum Master's activities.

3.2. Data Collection

We interviewed 28 software practitioners, distributed as follows: 19 developers, 4 designers, 3 testers, 1 project leader, and 1 manager. The population of our study was defined according to the four projects that were under development at the time of execution of the study. Furthermore, these practitioners were selected as interviewees because they faced daily activities that allowed knowledge creation.

Before starting to collect data, a researcher defined an initial questionnaire with open questions to guide the interviews. Then, a senior researcher analyzed the questions. Finally, the questionnaire was applied in a pilot interview. Table 2 shows some of the questions applied in our case study.

Using the results from the pilot interview, we were able to verify and to adequate the questions for the case study. Interviews took place in January, 2014 and were conducted in the participants' offices. All interviewees were informed of their volunteer participation and signed a consent form stating that they could withdraw from the research at any given time. The total time spent conducting the interviews was 8 hours and 25 minutes. Next, the interviews were transcribed and prepared for analysis in the Atlas.TI software (http://www.atlasti.com).

3.3. Data Analysis

To analyze the data collected, we used Grounded Theory (GT) techniques [9]. GT is a qualitative research method that uses a set of systematic data collection and analysis procedures to generate, prepare, and validate substantive theories on essentially social phenomena, or on wide social processes. The essence of the GT method is that a substantive theory emerges from the data. Thus, GT allows for producing a theory derived from systematically collected and analyzed data. Although the purpose of the GT method is the construction of substantive theories, its use does not necessarily needs to remain restricted to researchers who only have such research goal. Strauss and Corbin [9] explain that a researcher may use only some of its procedures to meet her research goal, e. g., when researchers need to understand some phenomenon.

The proposed GT coding process is split into three stages: open, axial, and selective [9]. Open coding involves the breakdown, analysis, comparison, conceptualization, and categorization of the data. In the early stages of the open coding, the researcher explores the data with a detailed examination of what is deemed as relevant through the intensive reading of the texts. Later, in the open coding stage the incidents or events are grouped in codes via incident-incident comparison. We used the Atlas.TI software to perform the open coding of the interviews. We randomly chose one of interviews as a starting point. While we analyzed the data contained within the interviews, we created codes associated with parts of the text.

In the axial coding, the purpose is to group the codes according to their properties-forming concepts that represent categories. Also, it is possible to identify categories' variations [21]. These categories are analyzed and subcategories are identified aiming to provide more clarification and specification. These subcategories can be dimensions of this category. The dimensions represent the attributes of a category along a continuum [9]. Finally, the categories and subcategories are related to each other, and the causal relationships between the categories are determined. We created categories according to the performed analysis of the data and we followed the purpose of axial coding step.

During the selective coding step, the goal is to perform a process refinement, identifying the core category, which will be related to all others. The core category should be able to integrate all other categories and to express the social process essence [9]. We decided not to elect a core category just yet. GT suggests that there should be interaction between the collection and analysis stages until the theoretical saturation is reached [9]. Consequently, we decided to postpone the selective coding phase. This is the main reason why we claim that we did not apply entirely the GT method, but only some of its specific procedures.

Table 2. Questionnaire sample

Question Type	Questions
Organizational Environment	How is your daily work here at the organization?
	If you have an activity that you don't know how to carry out, where do you find knowledge for performing it? How is this done? Does this happen frequently?
Knowledge Management	How do you identify important knowledge/lessons learned to be shared in the organization?
	When do you identify knowledge/lessons learned that would be relevant for your co-workers? How do you identify that? Alone, in a meeting, something else?
Organizational Learning	How did you learn how to carry out the activities of the organization?
	Which mechanisms are applied to stimulate the learning of the development process?

4. Study Results

We present our results organized by the three main categories identified during our data analysis process, namely: general knowledge sources, architectural framework, and lessons learned. Table 3 summarizes these categories. The types of knowledge are: (i) technical – knowledge regarding software routines and components; and (ii) software development specific – knowledge regarding software process and activities. More details about categories are described below.

Table 3. Categories of our qualitative analysis

Main Category	Description	Knowledge Type
General Knowledge Sources	Knowledge basis for supporting software development activities	Technical and Software Development
Architectural Framework	Source code repository for specific software products	Technical
Lessons Learned	Team and organizational aspects that can be knowledge for future projects	Technical and Software Development

4.1. General Knowledge Sources in the Organization

A general knowledge source is any location where the practitioners can obtain information from that helps them carry out their activities. Such sources also aid in the storage of the knowledge that was created during the execution of the project. We list some of the knowledge sources we identified below.

4.1.1. Written Material and Trainings. In our study, we considered books as written material. Written material and trainings provide important concepts and basic knowledge on technologies and the execution of the software development activities. Internal practitioners or consultants provide trainings. Such trainings can be designed to attend current needs of a project or to prepare the practitioners to play specific roles in projects to come. Trainings can also be conducted during an organizational event, known as 'Feature Friday'. During that event, a set of practitioners gathers and makes presentations of technologies or interesting results from other projects. Such event allows the knowledge socialization to occur. The following quotations present examples of such knowledge sources:

> *"(...) at the beginning of each year, we have a meeting with our Line Manager who defines how is our career today and what we define for the end of the year. (...), I focus my career in the management area. Therefore, my Line Manager can start to develop a career plan for me from my own wish to pursue that area, and then I can attend trainings on topics that can help me achieve my final goal, which is continuing working with management. (...)" - Interviewee 4.*

> *"(...) The institute has a great advantage. It facilitates a lot of things, such as the acquisition of books [written material], or even courses [trainings]. So, let's say that at the time I entered the company, yeah, the institute acquired some books that I needed to (...) Such support from the company was important to provide the materials to carry out the development"–Interviewee 1.*

> *"(...) an event where you can present your projects, concepts and workshops. I think the goal is exactly that, share knowledge". – Interviewee 20*

4.1.2. Web, Organizational Blog/Forum and Wiki. Team members use the Web as a source of important knowledge to aid in the execution of their activities from the moment they are hired. The Organizational Blog/Forum and the organizational wiki contain knowledge about the products that have been developed or are under development. The difference between these sources is the way in which one can access the information. The Organizational Blog/Forum has open access to anyone who has access to the Web while the Wiki is for internal use in the organization. We identified that the "wiki" is an important source for practitioners to seek for knowledge a colleague who has left the company used to hold. Moreover, we identified, in our qualitative analysis, a variation with respect to the consult activity of the wiki. Such variation is: the wiki is consulted or not consulted by practitioner. The reasons that led to such variations are: the lack of notification when there is an update in the wiki and the superficiality of the described content. The following quotation illustrates these knowledge sources.

> *"(...) then, the first thing I do is that: I go to the web, searching in a group, in that case, the Organizational Wiki." – Interviewee 3.*

> *"(...) maybe because they don't know that the information is being put there [Wiki]. (...) I have written papers inside the Wiki that very little people have seen because they don't know that I put them there. There is no dissemination or an effort to broadcast the results." – Interviewee 14.*

4.1.3. Experienced Software Practitioners. The experienced practitioners are a source of tacit knowledge. When there are questions regarding a certain technology or the software development process activities, the others practitioners often appeal to the more experienced practitioners. The

organizational environment can be an aspect that influences such tacit knowledge sharing between the practitioners since they work in the same office. The following quotations, from interviewees 6 and 11, illustrate such knowledge sharing:

"Question: When you have a difficulty in any of those activities, what do you do? Answer: I go after someone who has experience on that."

"(...) if I have a question, I contact the person who is responsible for that specific topic. Normally, it is a person who masters that specific topic."

There is no consensus on who should be contacted to obtain specific knowledge. When interviewed, the practitioners informed that they knew who was a specialist according to their previous projects, or they asked someone in the organization who to go to.

4.1.4. Source Code. The source code contains technical knowledge regarding the developed products within the organization. Source codes possess knowledge that is a legacy from other development teams and the client herself.

"(...) You made the software X that contains SMS application, do you have the source code? Do you know how it works?" - Interviewee 10.

"He [the project manager] gave me the source code. Then, he explained what he had done, using the source code itself." – Interviewee 18.

In certain software projects, a great amount of knowledge embedded within that source code is included in an architectural framework that contains parts of the developed software. Such architectural framework is described in the next subsection.

The presented knowledge sources can help the organizations to obtain general information regarding technologies and software development activities. Software practitioners have to use the available sources according to their needs. While the general knowledge sources are regarding to knowledge basis for software activities and technologies, the two next categories are more related to projects and activities executed to development software.

4.2. Architectural Frameworks

The project manager is typically responsible for a set of projects that possesses a similar architecture. Therefore, aiming at reducing rework, common knowledge to the projects of the same manager is added to architectural frameworks. These frameworks are created to standardize the development of applications and architectures. The standardized structure allows a better understanding of the knowledge contained within these frameworks.

Such frameworks contains only technical knowledge, i.e., it contains: (1) the source code – allows its constant maintenance and is a source of technical knowledge; (2) comments of the implemented solutions – allows other practitioners to understand more easily the code, thus reducing the negative impact in the development; and, (3) architectural decisions – decisions made by the projects team regarding the projects architecture.

"(...) [that framework] is updated. In fact, that is what I was doing right now. Since we are in a time when there is no project's tasks, we use the exceeding time to perform these improvements and generate the framework documentation. Actually, I was gathering things that were well performed in another project and trying to put that into the framework." – Interviewee 3.

"(...) at the beginning of this year, a huge project began with several small projects beginning in a way. Then, the small projects end, and then we have a short amount of time to gather and join everything into the framework. After that, we work in the next cycle of projects, which already uses what was previously created before. Finally, we arrive to the end of the year with a very robust framework, which is stored in our repository." – Interviewee 4.

The architectural frameworks can also assist distributed development teams. In certain situations, the client already possesses an initial framework of a project. Such framework contains codes that are necessary for developing the project.

"I once worked in a project where they, I mean… it was here in Manaus and people in Boston were involved, so we... they sent us code, the framework that they created there. Then, we had to understand and use what they had already created."

4.3. Lessons Learned

During the retrospective meetings the practitioners identify lessons learned regarding the projects. In the studied organization, such lessons are divided into three categories, namely: positive facts, negative facts, and improvement opportunities. Positive facts describe what happened that was good during the execution of a project. Negative facts are events that constrained the execution of the project, while improvement opportunities are things that worked well but could have been better. The identified negative facts and improvement opportunities are associated with action items since they offer the chance for improvements.

There is also the context of the lesson learned. Some lessons are related to the organization as a whole and some are related to the development team. Lessons learned that should be implemented in the organization as a whole are shared with the managers

by the Scrum Master. Moreover, lessons learned regarding the team are related to the improvement of the collective work.

> *"Yeah... at first, we carry out a review on the Sprint. Then, we will analyze what was good, bad, and then, what is related to the organization and to the team. Based on that, we make an action plan, trying to solve the things that went badly." – Interviewee 7.*

> *"(...) We classified what was bad for the team and bad for the organization." – Interviewee 3.*

After the meetings, the Scrum Master or the assigned person documented the lessons learned. We present next how accessible the lessons learned are to the team members and how the lessons learned management are handled by the project manners. This analysis is important because some concerns regarding such availability and management of lessons learned influence directly the knowledge creation and loss.

4.3.1. Availability of the lessons learned. We identified that there are distinct perceptions about the availability of the lessons learned. This maybe because the Scrum Master is the one deciding how to handle and make available these lessons to the team. Table 4 presents such subcategory proprieties, showing the variation of this subcategory [9]. In this case, the variation is about the frequency in which software developers access the lessons learned. Excerpts below illustrate the respondents' perception about the lessons learned availability:

> *"Do you have access to that material? I mean, that document [the lessons learned registration]? Answer: I don't know, I never accessed it, "oh, can I see?" well, I don't know if I had an easy access to that information. I don't know what to tell you."- interviewee 10.*

> *"(...) Do you have access to those facts [lessons learned]? Answer: Yes, he [the Scrum Master] provides them.*
> *- Where, exactly? Answer: We have a shared folder." - interviewee 6.*

> *"Is it available to you for consultation during the Sprint? Answer: I think it is available, but not that accessible since it is a document that is sent to the project manager." - interviewee 27.*

Table 4. Properties of "Availability of the lessons learned".

Category	Availability of the lessons learned
Concept	The way how practitioners access the lessons learned created in software projects
Variation axis	Positive: Accessing a lot Positive: Accessing a little Negative: Not accessing Negative: Not aware of the existence of the lessons learned

4.3.2. Management of the lessons learned. The most important person in charge of managing the lessons learned is the Scrum Master. Some Scrum Masters even insert the activities that were defined in the action items created to address the lessons learned into the project schedule. However, such activities can become implicit for the practitioners. Other Scrum Masters present to the project team what needs to be improved. Such Scrum Masters do not share the file that contains the lessons learned. This way, we identified the variation of management of the lessons learned subcategory: (i) scrum master does not provide the lessons and (ii) scrum master provides the lessons. Such variation was defined based on the following quotations.

> *"(...) the actions items end up becoming part of the kanban board of the Scrum Master. There is also a "mini kanban board" in our board that has our constrains, action items and extras." – Interviewee 5.*

> *"- Is the information that is shared during the retrospective meeting stored somewhere? Answer: Yes, it is. The Scrum Master stores all the suggestions that we put there.*
> *- And do you have access to that information? Answer: Hum, no. No.*
> *- How is that you remember which are the things that should be improved in the next Sprint? Answer: The Scrum Master show us and then we say 'Oh, OK'. They also discuss with the manager and normally nobody has access to those things. (...) It stays too dependent of the Scrum Master." – Interviewee 23.*

We observed that the way in which the Scrum Master manages the lessons learned can influence the access to those lessons learned by the practitioners. Figure 1 presents the relationship between the subcategory "availability of the lesson learned" and "management of the lesson learned" and their variations.

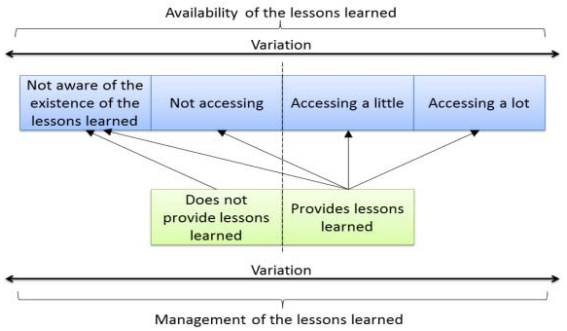

Figure 1. Relationship between subcategories

5. Discussion

In this section, we discuss our findings regarding knowledge creation and loss. This way, it is possible

to understand how our findings can contribute to knowledge creation and avoidance of knowledge loss. Additionally, we compare our findings with previous literature. Researchers should constantly compare theory and data in order to contribute for a better comprehension of their findings [22]. Such comparison aims at verifying how literature relates to the emerging data from this research [23].

5.1. Knowledge creation and loss

Knowledge sources can aid in the creation of knowledge within the organization. The Web and the written material document knowledge that is obtained from other sources. Thus, it is not possible to analyze the creation of such sources. The trainings facilitate the creation of knowledge for practitioners, since practices can be carried out in order to fixate the transferred knowledge. These trainings can complement both technical and software development knowledge. Training is important for allowing one to access knowledge [24]. Moreover, Kukko and Helander [16] found that lack of training is a barrier related to technological issues. This barrier can constrain the creation of necessary knowledge for the elaboration of products and the execution of software processes. In trainings, the presentations do not always remain available to the entire organization. Therefore, the collaborators who do not participate in the event might not have access to the knowledge that was created/shared through the discussion during the presentations.

The wiki also allows the knowledge creation. However, we noticed that the information within the wikis is superficial. We identified a variation point related to the creation/update of the knowledge inside the wiki. We verified that the update will depend on the development team. There are projects in which only the leader updates the wiki, and others in which we can see that the majority of the other practitioners carry out the updates. However it is not a common organizational practice. Thus, knowledge from the other team members can be lost at end of the some projects. It is also necessary to check the degree of utilization and update of wikis. Ras [25] found that knowledge embedded in wikis is related to the tools and processes of software engineering.

Some practitioners tend to ask experienced practitioners for further information. These experienced practitioners possess specialized knowledge on certain topics. This way, such experienced practitioners support the knowledge creation. We notice that these experienced practitioners are the key players for knowledge socialization. The usage of experience practitioners has strong relations to exploit knowledge [26]. In our findings, we identified that people look for experienced practitioners within the organization that could aid in the execution of their activities. The knowledge is retained within the practitioners who participate in the socialization. However, performing the externalization of all created knowledge through socialization with experienced practitioners can be very costly for the organization. Moreover, according to our findings presented in Section 4, it is possible that externalized knowledge is not used. Finally, the source code aids in knowledge creation/update.

Knowledge generated by architectural framework can aid in the knowledge creation and its maintenance for the organization. A body of knowledge that is common for that set of projects is then created. The creation and maintenance of knowledge through the architectural framework are also activities performed by certain practitioners. The context derived from this framework is related to the explicit and technical knowledge. In the literature, the source code aids in explicit knowledge learning. For instance, Faegri et al. [27] described a source code that aids new developers to understand the product that the organization develops. Anquetil et al. [28] identified that thoroughly analyzing the source code demands cost and effort. Our results show that by following the procedures/standards for maintaining the architectural framework do facilitate the software development and knowledge flow for other practitioners who will use the framework. This way, such framework improves the organization's productivity and avoids knowledge loss.

Despite the fact that some frameworks possessed knowledge that came from what worked well in other projects, only a group within the organization has permission to contribute to the creation of new knowledge. Just who is part of the architecture group has access to the framework. In our investigation, only three interviewees commented on creating knowledge in the frameworks. Due to the low number of practitioners who contribute to the evolution of the frameworks, relevant technical knowledge can be lost.

In definition of lessons learned, we identified that such lessons allow the creation of both technical knowledge and knowledge on the software process. Nonetheless, it is necessary to adequately handle these lessons learned in order to avoid their loss. Passos et al. [29] found that the reuse of experiences and lessons learned can assist the improvement of the project results in terms of time productivity and product quality.

Regarding knowledge loss in lessons learned, we noted that not always the action items are provided to

the team members. Despite knowing that the lessons learned are being documented, some teams do not know what is actually being done with that knowledge. Such lack of dissemination of the lessons learned can make the practitioners forget what can be improved in the next project/sprint. Additionally, the lack of knowledge on the goal for reporting lessons and their access can lead to the loss of knowledge and can cause problems to repeat in other projects.

5.2. Implications for Industry and Research

The results of our exploratory case study have implications for both practitioners and research. The execution of an exploratory case study allowed us to analyze what is going on in the organization regarding the creation and loss of knowledge of the developed products, technologies, and activities within the development process. Our results show how knowledge is created in practice and issues that can lead to the loss of knowledge. This understanding allows us to better prepare the available knowledge that is provided in software organizations in case newcomers enter the organization or experienced practitioners leave it.

Practitioners must consider that our case study is not able to identify all knowledge creation and loss issues. However our findings can be used as a starting point to analyze and improve the identified issues in other software organizations. Practitioners must bear in mind that each context has its specific aspects and therefore knowledge creation and loss issues must be adapted to fit the organizations' expectations within a determined context. Our outcomes and those in the literature can help future organizational knowledge practices, particularly, in similar organizations.

We also identified insights and new ideas for future research such as the analysis of the technical knowledge embedded within the source code of the architectural framework, in order to classify that knowledge and instantiate it in the organization. Furthermore, an action-research can be performed with the aim of minimizing issues regarding knowledge loss within the organization.

6. Final Remarks

Exploratory case studies are carried out to verify what is going on in the real world and generate insights and ideas [18]. In this paper, we report on an exploratory case study about knowledge creation and loss in a software organization. Our data collection and analysis were based on semi-structured interviews and GT techniques, respectively.

The major contribution of this paper is the analysis, in industrial setting, of three main topics that influence knowledge creation and loss in a software organization. First, the knowledge sources that provides a way for creating and maintaining basic organizational knowledge. Second, the architectural frameworks that possess source codes, comments, and architectural decisions on a set of projects. The framework is maintained by the organization. Therefore, the technical knowledge contained in this framework is always updated. The last topic that we identified in our results is regarding lessons learned. In this sense, lessons learned handle knowledge that is not supported by the frameworks, such as improvement opportunities for the teams and issues regarding the execution of the software processes. The results obtained with this case study can be used to guide software organization on how to improve their practices regarding the creation of knowledge. KM within the organization can be adapted, aiming at reducing issues regarding knowledge loss. Furthermore, it can standardize the knowledge creation activities during the retrospective meetings. This way, the organization can avoid the knowledge loss.

Our study has some limitations regarding its participants. Some practitioners might have answered differently if they felt they were being evaluated by the researcher during the interviews. This behavior can bias our findings, since a practitioner can say what he/she thought was more appropriate instead of the truth. In order to mitigate this threat to the validity of our results, we provided a consent form explaining that the study aimed at obtaining information regarding the organization and not to evaluate the performance of the practitioners. Also, we informed the practitioners that all personal data would remain confident. Additionally, our data collection involved practitioners in a usual software working environment. This way, we selected a natural setting required by the case study approach. We also know that generalization is a limitation when one studies a single case. Despite this limitation, we believe that our study offers relevant contributions to literature since it aids to advancing the state of art in the topic, providing evidence that can be later tested using quantitative methods. In short, our study helps to build a body of knowledge about knowledge creation and loss. It is important to observe what happens in a real software development context in order to contribute for improving the organizational knowledge practices.

Our next step will be to replicate this study in other software organizations with the purpose of corroborating our findings. If possible, we also intend

to identify further issues regarding the creation and loss of knowledge within software organizations.

7. Acknowledgments

The authors would like to thank the INDT. The first author would like to thank Luis Rivero for his remarks on this paper. Also, we would like to thank the support granted by FAPEAM through processes: 062.00146/2012; 062.00600/2014; 062.00578/2014; 01135/2011; Edital 009/2012 - RHTI Doutorado.

8. References

[1] Levy, M., and Hazzan, O., "Knowledge Management in Practice: The Case of Agile Software Development". in ICSE Workshop on Cooperative and Human Aspects on Software Engineering, 2009. CHASE '09. Vancouver, 2009, pp. 60-65.

[2] Mendonça, M.G., Seaman, C.B., Basili, V.R., and Kim, Y.-M., "A Prototype Experience Management System for a Software Consulting Organization", in Proc. of the SEKE 2001. Buenos Aires, AR, 2001, pp. 29-36.

[3] Begel, A., and Simon, B., "Novice Software Developers, All over Again", in Proc. of the Fourth international Workshop on Computing Education Research, ACM, 2008, pp. 3-14.

[4] Sundaresan, S., and Zuopeng, Z., "Facilitating Knowledge Transfer in Organizations through Incentive Alignment and It Investment", in Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004, pp. 10 pp.

[5] Nonaka, I., Toyama, R., and Konno, N., "Seci, Ba and Leadership: A Unified Model of Dynamic Knowledge Creation", Long Range Planning, 33(1), 2000, pp. 5-34.

[6] Dingsoyr, T., Bjornson, F.O., and Shull, F., "What Do We Know About Knowledge Management? Practical Implications for Software Eng.", IEEE Software, 26(3), 2009, pp. 100-103.

[7] Ruhe, G., "Software Engineering Decision Support – a New Paradigm for Learning Software Organizations", in (Henninger, S., and Maurer, F., 'eds.'): Advances in LSO, Springer Berlin Heidelberg, 2003, pp. 104-113.

[8] Mitchell, S.M., and Seaman, C.B., "Software Process Improvement through the Identification and Removal of Project-Level Knowledge Flow Obstacles", in Proc. of the ICSE, 2012, pp. 1265-1268.

[9] Strauss, A. and Corbin, J., Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, Sage, 1998.

[10] Chau, T., Maurer, F., and Melnik, G., "Knowledge Sharing: Agile Methods Vs. Tayloristic Methods", in IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003, pp. 302-302.

[11] Schneider, K., Experience and Knowledge Management in Software Engineering, Springer Heidelberg, 2009.

[12] Bjørnson, F.O., and Dingsøyr, T., "Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts, Findings and Research Methods Used", Information and Software Tech, 50(11), 2008, pp. 1055-1068.

[13] Aurum, A., Daneshgar, F., and Ward, J., "Investigating Knowledge Management Practices in Software Development Organisations – an Australian Experience", Journal of Information and Software Tech., 50(6), 2008, pp. 511-533.

[14] Desouza, K., Dingsoyr, T., and Awazu, Y., "Experiences with Conducting Project Postmortems: Reports Vs. Stories and Practitioner Perspective", in Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS '05., 2005, Island of Hawaii (Big Island) pp. 233-233.

[15] Dingsøyr, T., Moe, N., Schalken, J., and Stålhane, T., "Organizational Learning through Project Postmortem Reviews – an Explorative Case Study", in (Abrahamsson, P. et al., 'eds.'): Software Process Improvement, Springer Berlin Heidelberg, 2007, pp. 136-147.

[16] Kukko, M., and Helander, N., "Knowledge Sharing Barriers in Growing Software Companies", in Proc of the 45th HICSS, 2012, Maui, Hawaii. pp. 3756-3765.

[17] Yin, R., Case Study Research: Design and Methods, Sage, Beverly Hills, 2009.

[18] Runeson, P., and Höst, M., "Guidelines for Conducting and Reporting Case Study Research in Software Engineering", J. of Emp. Software Engineering, 14(2), 2009, pp. 131-164.

[19] Seaman, C.B., "Qualitative Methods": Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 35-62.

[20] Schwaber, K. and Beedle, M., Agile Software Development with Scrum, Prentice Hall, USA, 2002.

[21] Glaser, B.G., and Holton, J., "Remodeling Grounded Theory", in Forum: Qualitative social research, 2004

[22] Taipale, O., Karhu, K., and Smolander, K., "Observing Software Testing Practice from the Viewpoint of Organizations and Knowledge Management", in Proceedings of First International Symposium on Emp.Software Engineering and Measurement (ESEM 2007). 2007, Madrid, Spain. pp. 21-30.

[23] Hoda, R., Noble, J., and Marshall, S., "Using Grounded Theory to Study the Human Aspects of Software Engineering", in Proceedings of Human Aspects of Software Engineering (HAoSE '10), 2010, Nevada, USA. pp. 1-2.

[24] Newman, I., "Observations on Relationships between Initial Professional Education for Software Engineering and Systems Engineering-a Case Study", in Proceedings of 14th Conference on Software Engineering Education and Training, 2001, New York, USA. pp. 172-181.

[25] Ras, E., "Investigating Wikis for Software Engineering - Results of Two Case Studies", in Proceedings of ICSE Workshop on Wikis for Software Engineering (WIKIS4SE '09), 2009, Vancouver, Canada. pp. 47-55.

[26] Ehrlich, K., and Chang, K., "Leveraging Expertise in Global Software Teams: Going Outside Boundaries", In Proc. of International Conference on Global Software Engineering (ICGSE), 2006, Florianopolis, Brazil. pp. 149-158.

[27] Fægri, T.E., Dybå, T., and Dingsøyr, T., "Introducing Knowledge Redundancy Practice in Software Development: Experiences with Job Rotation in Support Work", Information and Software Technology, 52(10), 2010, pp. 1118-1132.

[28] Anquetil, N., De Oliveira, K.M., De Sousa, K.D., and Batista Dias, M.G., "Software Maintenance Seen as a Knowledge Management Issue", Information and Software Technology, 49(5), 2007, pp. 515-529.

[29] Passos, C., Braun, A.P., Cruzes, D.S., and Mendonça, M., "Analyzing the Impact of Beliefs in Software Project Practices", in Proc. of the ESEM, 2011, Banff, Canada. pp. 444-452.