Knowledge Transfer between Senior and Novice Software Engineers: A Qualitative Analysis

Davi Viana and Tayana Conte
USES Research Group
Universidade Federal do Amazonas
Amazonas - Brazil
{davi.viana, tayana}@icomp.ufam.edu.br

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

Abstract— Software development is a knowledge intensive activity. Software engineers need to gather domain knowledge to be able to successfully deliver a software system. In particular, novice software engineers need to acquire enough knowledge to perform their tasks. This means that knowledge transfer to novice software engineers must be quickly and effectively performed to facilitate the onboarding process. One way to understand the knowledge transfer process is by analyzing the software development context and the involved team members. Such analysis enables the development team to determine key aspects that can influence knowledge acquisition by novice software engineers. This can also allow the identification of possible strategies that minimize the effort employed by the team members during this process. This paper presents a qualitative study about knowledge transfer in a small software organization from the point of view of software practitioners. Our results suggest that software developers have several knowledge sources and novice engineers learn when they observe organizational procedures and when the tasks have detailed guidelines. In addition, we identified that developers carry out most steps of the knowledge creation process defined by Nonaka and Takeuchi. We believe our results can support other software organizations to improve the sharing of knowledge and learning practices.

Keywords-component: Onboarding, Knowledge Transfer, novice software engineers, qualitative analysis.

I. INTRODUCTION

In software development activities, dealing with knowledge can be particularly critical, since these activities depend heavily on knowledge and individuals [1, 2, 3]. Novice software engineers must have knowledge about: (1) technologies; (2) software engineering (SE) methods; and, (3) the organization’s internal processes [4]. Nevertheless, carrying out activities that involve such amount of knowledge is not trivial [5]. In order to act according to the market’s demands, knowledge must be analyzed and shared among all employees [2, 6].

Nonaka et al. [7] classify knowledge into two types: tacit and explicit. Tacit knowledge is based on the person’s experience and is hard to formalize. On the other hand, explicit knowledge, also called codified knowledge, is considered to be transferable using a formal and semantic language. Nonaka et al. [7] also state that explicit knowledge can be represented in documents and databases. Furthermore, explicit knowledge must be adequately structured so that other team members are able to use it. Despite the type of knowledge, tacit or explicit, it is necessary to manage this knowledge since knowledge is considered to be the main competitive asset of a software organization [3].

According to Wang et al. [8], traditional training models often draw on limited resources that are insufficient to support current training efforts. This question can be more severe when organizations need to train and transfer basic organizational knowledge to novice software engineers. Therefore, it is important to understand how novice software development practitioners learn the relevant knowledge on their organizations. Knowledge creation and transfer are inseparable from learning in an organization; both knowledge activities are in fact the consequence of learning [9].

The goal of this paper is to analyze, through a qualitative empirical study, how novice software engineers gain knowledge during the initial activities they perform in a software organization. We also analyze which factors influence such knowledge transfer.

In order to better understand such factors, we performed a qualitative study based on interviews. Such study was performed in a software development organization that encourages knowledge sharing in the work environment. Such sharing was important because some of the senior engineers were going to leave the organization. When talking about novice practitioners we refer to beginners who are learning how to perform assignments in a team. The lack of defined software processes makes knowledge transfer and sharing more complex in the studied organization. Thus, the observation of knowledge transfer practices in this organization is important because we can use our findings to verify the nature of such practices in a real context.

Our results show that software developers in this organization have several knowledge sources including the web, the intranet, training material, the source code, expert team members and the software documentation. However, novice practitioners usually prefer to use informal conversations, which are allowed in the organization’s environment. In other words, our results illustrate a real process of knowledge transfer and learning software development activities. The understanding of how software knowledge transfer occurs can support the creation of new strategies for improving novice engineers’ capabilities (e.g. stimulus for using the knowledge sources).

The rest of this paper is structured as follows: Section 2 presents related work regarding knowledge transfer in SE.
Section 3 describes the defined research method, the study settings and study context. Then, in Section 4 we discuss our findings, a comparison between our results and findings from literature, and a discussion of our results and the knowledge creation theory defined by Nonaka and Takeuchi [10]. Finally, Section 5 presents our conclusions and future work.

II. RELATED WORK

Software development is a knowledge intensive activity. For instance, software engineers need to gather domain knowledge to be able to successfully deliver the software system. In addition, novice software engineers need to acquire enough knowledge in order to perform their assignments. Furthermore, not all team members have enough knowledge for performing their role [11]. In this sense, it is necessary to stimulate the transfer of knowledge among software development members. According Joshi et al. [12], knowledge transfer occurs when knowledge is broadcasted from one entity (e.g., an individual) to others.

Many researchers aim to propose and analyze strategies for transfer the all necessary knowledge in a software organization. Aurum et al. [13] performed a qualitative study in two Australian organizations and concluded that team members believe in the utility of sharing knowledge. However, the teams’ ability of using Knowledge Management (KM) mechanisms was limited. Furthermore, the authors observed that team members created informal networks, and when supporting such networks with KM, it was necessary to formalize the practices of knowledge sharing. Finally, they identified that the tools, techniques and methodologies used in the organizations were inadequate for an effective knowledge management and transfer.

In order to analyze novice practitioners, Begel and Simon [4] performed a case study at Microsoft. The authors analyzed all aspects of novice practitioners’ jobs: coding, debugging, designing, and engaging with their team; and analyzed the types of tasks in which they engaged. In that research, novice practitioners showed high capability in the functional and technical job aspects, but lack of preparation and training in the social and communication aspects they faced at a daily basis [4]. The authors believe that novice practitioners’ naivety caused stress and poor productivity during their first months.

Regarding tools for sharing and transferring software knowledge, Amescua et al. [14] describes the usage of wikis for storing software engineering best practices. Such research demonstrated that wikis facilitate an effective learning environment. They help users learn an agile process through mechanisms for knowledge sharing, and provide a repository with useful artefacts. Moreover, the authors concluded that novice software engineers could develop software products more independently.

One way to integrate both tacit and explicit knowledge is presented by Sandhawalia and Dalcher [15]. They propose a framework to integrate both types of knowledge and to facilitate the flow of knowledge to address unstructured situations in software projects. They verified that knowledge flow depends on the organizational area.

Although the literature presents numerous researches on knowledge sharing in software engineering, it is important to observe and to analyze how knowledge transfer occurs during project execution with undefined software processes. Such analysis can provide insights to support a successful knowledge transfer and learning by novice software engineers.

III. RESEARCH METHOD AND STUDY SETTING

One way to investigate the software developers’ point of view is to use qualitative methods. According to Seaman [16], the use of qualitative methods allows the researcher to consider human behavior and thoroughly understand the studied object. Therefore, by performing a qualitative study, we hope to obtain a more adequate understanding of the process of knowledge transfer when performed by novice practitioners.

We collected data for this qualitative study using three interviews with a semi-structured questionnaire. First of all, we prepared a questionnaire with open questions about knowledge sharing and the organizational environment. Such questionnaire was applied in the first interview with an expert software engineer. After that, we transcribed and analyze the interview using grounded theory techniques [17]. This analysis helped us adapt the questionnaire to the organizational context and according to the novice software engineers' perspective. Finally, we conducted and analyzed two additional interviews with novice engineers and drew our conclusions. More details about the organization, data collection and analysis methods are presented below.

A. Study Context

The interviews took place in a software organization called Beta located in Manaus, a city in the North of Brazil. Such organization needs to share knowledge in its different projects in order to increase the new employees’ abilities. Beta is a small (10 employees) software development company within the customs control agency and develops software for supporting such agency in the management of goods flow in the Free Economic Zone of Manaus.

Beta adopted agile method in the past. However, it currently has an undefined software process. Team members receive the assignments according to the customs agency needs. Due to the lack of a defined software process, the observation of knowledge transfer practices in this organization is important because we can verify the nature of such practices.

B. Data Collection

For the interviews, semi-structured questionnaires with open questions were prepared. We prepared two types of questionnaires: one for the expert and another one for novices. Table I shows some of the questions asked to novices.

<table>
<thead>
<tr>
<th>Knowledge transfer activities identification</th>
<th>How do you identify important knowledge to share in your project?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge characterization</td>
<td>How do team members share knowledge in your organization?</td>
</tr>
<tr>
<td></td>
<td>When you identify important project knowledge, which information do you try to get?</td>
</tr>
<tr>
<td></td>
<td>How can this knowledge help you during the software project?</td>
</tr>
</tbody>
</table>
Our interviews took place in the interviewees' offices at the Beta organization and the researcher did not rush the interview so that the interviewee could express his/her opinion effectively.

C. Data Analysis

To analyze the data obtained through the interviews, we used Grounded Theory (GT) techniques [17]. 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 the substantive theory emerges from the data, that is, it produces 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 need to remain restricted only to researchers who have this research goal. According to Strauss and Corbin [17], the researcher may use only some of its procedures to meet his/her research goals. That is our case: we did not use all the process suggested by GT, but focused on the coding stages because those were the relevant ones for our study.

The coding process can be split into three stages: open, axial, and selective coding [17]. Open coding involves the breakdown, analysis, comparison, conceptualization, and the categorization of the data. In the early stages of open coding the researcher explores the data with a detailed examination of what one deems as relevant due to the intensive reading of the texts. In the open coding stage the incidents or events are grouped in codes via the incident-incident comparison. In the next step, axial coding, the codes are grouped according to their properties forming concepts that represent categories. These categories are analyzed and subcategories are identified aiming to provide more clarification and specification. Finally, the categories and subcategories are related to each other, and the causal relationships between the categories were determined.

We conducted two periods of data collection and analysis. Each analysis was held after each data collection phase. After each analysis, another researcher verified the identified codes and the created categories in order to audit the coding process. Our informants (i.e., the practitioners within the analyzed organization that we interviewed) are representative since they validate the data we collected, and as a consequence of our results. Therefore, we have confidence in the accuracy and validity of the data we collected, and as a consequence of our results.

IV. STUDY RESULTS

In this section we discuss the results from our empirical study. Furthermore, we compare the findings from the qualitative analysis with the ones found in literature.

A. Knowledge Transfer at the Beta Organization

As mentioned before, Beta develops software within a customs control agency. Besides the lack of defined software process, there is not a formal way to transfer knowledge within the organization. Thus, the main way to analyze how knowledge transfer happens in practice is by verifying this process among senior and novice developers. In fact, practitioners have a commitment to transfer the organizational knowledge to novice practitioners because they believe such people will be the future expert members.

Developers at the Beta organization have several knowledge sources available: the Web, the intranet, training material, the source code, expert members and the software documentation. However, novice practitioners usually ignore the software documentation as a knowledge source and prefer informal conversations, which are allowed in the organization's environment. Novice engineers learn when they observe organizational procedures and when the assignments they are supposed to perform have detailed guidelines. Expert team members create such detailed guidelines due the lack of novices' expertise. Thus, both types of knowledge (tacit and explicit) are emphasized at the Beta organization.

Before beginning their assignments, novice practitioners had to participate in diverse trainings focusing on programming languages, database and business context. Such trainings are important for the onboarding process of novice engineers in the organization by providing basic required skills for novice's assignments.

After trainings, when novices start working, they receive new assignments every day. These assignments contain guidelines that they have to follow in order to successfully carry them out. A senior team member sends these assignments and guidelines by e-mail.

“We do not have a well-defined schedule, (...) I provide novices with what they need to do [their assignments]. Also, I give guidelines about what their tasks are. In the afternoon, after lunch, I talk to them again and I verify if they have any doubts and so on [until they finish the assignment]” - Senior Team Member Interviewee.

“I arrive and check my e-mail, because the senior member sends my assignments. So, if there is something I have to do, I start doing it. I am doing the assignments according to the instructions from the guidelines. If I have any doubts, I talk to the senior team member. – Novice Interviewee 2

During our study, we noticed that e-mails contained a lot of information (knowledge) regarding organizational assignments, such as programming functions that support the custom agencies' routines. Additionally, we noticed that the most detailed knowledge is shared face to face according to the organization’s needs.

“... How do you share relevant knowledge among the project team?
- I think that when demands are appearing. For example, when there is a new function that has to be developed, the senior member says: ‘we do this in a certain way, because the framework was defined a long time ago.’ The senior member explains how the framework works. So it is more a tacit knowledge.” - Novice Interviewee 1

Another way to obtain knowledge is through observations. Novice developers observe how other team members perform their assignments. When doing so, they perceive how the business process occurs. In this way, they acquire tacit knowledge.

“... How do you obtain this knowledge?
- I obtain knowledge through the observation of the organizational process and by analyzing what each part of the software does. Regarding programming, I observed the source code too.” - Novice Interviewee 2

We noticed that the senior developer provided a complete assignment guideline because he did not have high confidence regarding the novice engineer’s performance and expertise. Additionally, we noticed that novice engineers acquired more knowledge (about programming techniques and the business domain) when they performed the assignments by themselves. Although initial information had been provided, in most cases, it was necessary to search for additional knowledge sources that helped the assignments’ success.

Novice practitioners pointed out the following knowledge sources: (a) the web; (b) the intranet; (c) books; (d) training material; (c) expert team members; and (d) software source code. The following quotations present examples of such sources:

“Before asking someone, I try to solve my problem. If I do not know how to solve it, I consult some software program that I implemented. Most of the time, we develop, however we do not keep all in mind. Then, if I haven’t solved my problem yet, I search the web. Lastly, I ask for help from expert team members.” - Novice Interviewee 1

“(…) The software has some similar functions that I can use as examples. So, I develop based on such examples (…) If I have any doubts, I ask him [senior member] too (…). We have C# books that show some examples” - Novice Interviewee 2

Our informants told us about another knowledge source, the software documentation (e.g., requirements specification, UML diagrams, and software plans). The senior member stated that novice engineers are aware of this documentation. However, it is out of date and novice engineers did not mention its utilization.

“…The software documentation started, if I am not mistaken, in 2008. Then, it is probably out of date…” - Senior Team Member Interviewee.

During the execution of the assignments, novice engineers ask expert team members for help. We verified that all team members are in the same office. Thus, we concluded that the office configuration facilitates face to face interaction. Nevertheless, there are not knowledge transfer records. Then, the senior managers cannot know what knowledge was transferred.

The usage of the source code as a knowledge source is possible due to the organizational code standard. Such standard recommends the creation of relevant comments in the source code. We notice that code standard is very important, because it helps maintain the knowledge about the software when some team member stops working in Beta, for example.

“Usually, we create comments in the source code. We explain the code step-by-step. The organization has a standard for software development. Anyone who knows such standard can read the source code. This standard facilitates a lot.” – Senior Team Member Interviewee

One novice practitioner stated that some knowledge sources (e.g., books and the Web) are less used as the days pass. The most common source of knowledge is the expert team members. Fig. 1 shows the relations between the codes about knowledge transfer that we identified regarding the results presented above. In order to perform the open and axial coding of the interviews, we used the ATLAS.ti software.

We can see that the “knowledge transfer to novice engineers” category has three main associations: knowledge sources; observations of organizational procedures; detailing of assignments’ descriptions for novice practitioners. Each one contributes for the knowledge sharing process.

Regarding evaluation of the knowledge acquired by novice software engineers, the senior member evaluates this informally through the observation of the quality of the releases. This evaluation helps the senior team member increase the complexity of the activities assigned to the novice developers. Such increase will ensure that the novice practitioners acquire new knowledge and gain additional experience regarding the software and the organizational business model. While analyzing these results, we noted some aspects that can influence the sharing of knowledge. Table II presents such aspects and their description.

B. Comparison Findings

In this section, we discuss our current results compared to previous literature results and prior results described in our own previous work [18]. Some aspects reported by such publications are important and contribute for a better comprehension of our findings. The comparison of our results with these publications aims to verify how literature in the field relates to our emerging data [19]. Regarding our prior investigation, we performed another qualitative study about the life cycle of lessons learned in a software organization [18].
We consider lessons learned as a kind of knowledge artifact, i.e., an artifact that stores knowledge. In this study we observed some similar knowledge sources and practices. To avoid possible misunderstanding, we call "Alpha" the organization in our prior study [18]. Such comparisons give us more confidence on our results.

At Beta, senior members have a great commitment for improving the novice engineers' skills in order to integrate novice practitioners into the software team. Furthermore, we identified that knowledge is mainly transferred by face-to-face interaction. Knowledge transfer occurs during project development and in an informal way. This shows that the knowledge is handled more tacitly. Joshi et al. [12] stated that communication is a primary mechanism for transferring knowledge. The authors state that face to face communication is more significant than virtual communication.

In our previous research, we detected that, at Alpha, the knowledge is handled explicitly. Alpha adopted agile development methods and had an organizational culture to explicit its lessons learned. Nonetheless, they did not mention the use of code as knowledge source. Also, the software documentation in these projects is constantly updated [18]. At Beta, the software documentation was not updated. Yet, the source code had been standardized and it was used as a knowledge source. In future investigations, we plan to verify to what extent the source code and software documentation assist the activities related to knowledge acquisition.

Training is important because they allow one to access important knowledge [20]. Such trainings are important to address some issues, as lack of soft and technical novices' skills [21]. In our findings, novices and senior team members emphasize the importance of training because it presented an overview about the used technologies and the organizational business process. However, training does not cover complete knowledge about all the aspects of the organization.

At Beta, all team members are allocated in the same office. This environment facilitates tacit knowledge sharing. However, team members do not record this knowledge. Tsai and Cheng [22] stated that organizational climate (as the feeling conveyed in a group by the physical layout and the way in which members of the organization interact with each other) contributes positively to sharing both types of knowledge.

The interaction between tacit and explicit knowledge creates the SECI model (Socialization, Externalization, Combination, and Internalization) [10]. Such model supports the creation, exploration and maintenance knowledge. The Socialization is the process of sharing experiences, in which the person shares tacit knowledge directly with others through observations and collaboration. The Externalization converts tacit knowledge into explicit knowledge, thus allowing it to be shared with the project team. In the Combination process new explicit knowledge is created based on a combination of different explicit knowledge. The internalization process makes the conversion of explicit knowledge into implicit and finally maybe tacit knowledge. We compared the knowledge treatment practices in Beta regarding the SECI model [10], we noticed some activities related to each process, according to Figure 2.

The socialization process is highlighted due to the organizations' characteristics presented before, they are in the same office and novices can always talk to senior members. We can point out that an organization that converts tacit knowledge in explicit knowledge through detailed e-mails about work activities and commented source code can help explicit relevant knowledge about the software. Regarding the internalization process, novice practitioners receive the explanation of assignments by e-mail, they can consult similar software source code, intranet, books and Web. Nonaka and Takeuchi [10] state that explicit knowledge is shared in all organization and it is converted into tacit knowledge by individuals. This knowledge conversion is related to "learning

![Figure 2. Our findings and SECI Model.](image-url)
by doing”. Thus, we verified that when novice practitioners have finished their assignments, they are learning something new. Finally, our results do not allow us to identify any activities related to Combination process.

V. FINAL REMARKS AND FUTURE WORK

In this paper, we discussed knowledge transfer in a small software organization. The lack of usage of a defined software process allowed us to analyze knowledge sharing in the field, i.e., in the context of the Beta organization. A qualitative study was conducted seeking to understand knowledge transfer among senior and novice software engineers. In addition, it was possible to observe tacit and explicit knowledge dissemination.

We noticed that novice practitioners had an initial contact with the necessary knowledge to perform their assignments through training provided by senior engineers. However, our results suggest that they gained more knowledge during the actual software development activities. Also, as mentioned above, a successful knowledge sharing was important for the organization because some of its senior engineers were leaving. The informal development process contributed to an informal way of knowledge transfer. In addition, we observed that Beta adopted parts of the SECI model proposed by Nonaka and Takeuchi [10]. Our results can be used by other software organizations as recommendations for improving their knowledge sharing activities. Additionally, it is necessary to encourage the complete usage of tacit and explicit knowledge in order to create benefits for organizations.

This study has some limitations regarding its organizational context. We carried out interviews with novice and senior team members. We covered knowledge transfer among novices as a whole. However, we know that as a qualitative study, we cannot generalize our results to other companies. Despite that, we believe our study is important because each qualitative study contributes to advancing the state of art in a research area, providing evidence and hypothesis that can be later tested using quantitative methods. In short, each study helps to build a body of knowledge about the sharing of knowledge. Many studies have been performed in academic settings, while little studies have been conducted with novices on corporate contexts. It is important to observe what happens in a real software development context in order to contribute for improving the organizational practices.

We plan to perform additional qualitative data collections aiming to understand more organizations and other contexts. This will allow us to gain a better understanding about knowledge transfer and identify support strategies to organizational learning in software companies.

ACKNOWLEDGMENT

The authors would like to thank the team members of Beta organization. We would also like to thank Luis Rivero for his remarks on this paper. Also, we would like to thank the financial support granted by FAPEAM RHTI - Doctorate; Project N.021/2011 - FAPEAM Universal Amazonas; and FAPEAM process PAPE 032/2013

REFERENCES