# Challenges Faced in the Process of Knowledge Transfer in an Industrial Environment

Mauricio Soto
mauriciosoto@cmu.edu

Fall 2017

**Abstract**

Knowledge transfer in an industrial context from an existing enterprise to a new emerging branch is a slow and troublesome process. To become productive, there is a vast amount of knowledge expected to be acquired by the teammates in the emerging branch in a limited period of time; but the teammates in the emerging branch can only process so much knowledge in a limited time frame. By using a hybrid approach between face-to-face communication and proper documentation, colleagues at the new branch are able to grasp the required knowledge faster with minimum help from mentors. The knowledge transfer process between branches can therefore be optimized for the time restrictions and minimize the time it requires for the new branch to be ready for production.

# 1 Introduction

Opening new branches is a common step for expanding companies. This enables a pool of resources that enterprises would not be able to obtain otherwise, such as special talents, legal benefits, or financial gain [2]. However, opening a new branch is accompanied by a set of inherent challenges.

One of the main challenges in this process is transmitting the knowledge, culture, and mechanisms necessary for the coworkers in the new branch to perform their tasks adequately. This knowledge transfer process is particularly complex in the context of opening a new branch for two main reasons: First, there is an uncommonly large number of new coworkers joining the company through the new branch, usually in a distant location, all of which require domain specific knowledge. This is different from other more studied cases, such as transferring knowledge to a newcomer that joins an already existing branch. Second, there is a limited amount of knowledge that can be processed by the new coworkers in a particular time frame, and often the volume of knowledge required to be processed by the new coworkers is overwhelming for the limited time frame.

Domain specific knowledge is one of the key assets to maintaining competitiveness in software enterprises. To sustain competitive advantage, companies must assess their knowledge resources and establish their knowledge strategy [3]. In this context, the value of effective knowledge transfer becomes crucial.

I have had the opportunity to work in diverse contexts of knowledge transfer in an industrial environment. I will compare and contrast my experiences, with an emphasis on the latest experience I had before joining the PhD program, wherein a large, well-known enterprise opened a new branch in Costa Rica.

# 2 Knowledge Management and Transfer

Knowledge is one of the most important organizational resources that enterprises possess. The transfer of this knowledge, especially from senior to junior collaborators, is key to maintaining, augmenting, and making proper use of that knowledge base. *Knowledge management* is the process of dealing with the management of knowledge and its related activities. The main goal of these activities is to make the enterprise act as intelligently as possible to

secure its viability and overall success, and to realize the best value of its knowledge assets [12].

Organizing the process of passing down knowledge has noticeable gains, such as: financial value, operational benefits, business process improvement and culture [6]. These translate into concrete assets such as improvements in cost, quality, cycle time, lead time, decision making and resolved complaints.

Learning is tightly connected to the knowledge transfer process. There are two kinds of knowledge: When you learn **about** something, referred to as *explicit* knowledge; and when you learn **to do** something, called *tacit* knowledge [4]. *Cognitive learning* is when someone learns **about** a topic, as opposed to *behavioral learning* which refers to the act of learning **to do** something. In the case of knowledge transfer in a industrial environment, collaborators need both cognitive and behavioral learning.

When newcomers join a new software project, they typically find themselves in an unfamiliar project landscape. There are three primary factors that impact newcomers' success when facing a new project: early experimentation, internalizing structure and culture, and progress validation [5]. These three main factors have a strong connection to knowledge transfer.

Knowledge transfer in an industrial context can take mainly two forms: face-to-face interactions between one or several apprentices and a mentor, or documentation (i.e. internal comments in the code, external documentation, educational videos, directives, etc). Other practices that boost the quality of the learning process are, for example, to appoint mentors whom share recent contextual information and focus on related issues [10]. Providing support for selective information disclosure that maps to the newcomers' mental models becomes a crucial part of the learning experience. This enables collaboration between apprentice and mentor, and sharing of ideas and reflections, enhancing the overall learning process [9]. Constant interaction and geographical proximity between mentor and apprentice also enhance communication, as well as practices such as openness, honesty, trust, respect and transparency [11].

Documentation is especially important when face-to-face interaction is difficult to obtain. Documentation is particularly important for the software components that are intended to be reused the most [8]. Parnas mentions that even when software components are built using good design, he does not see them being reused if they are not well documented. [1].

In both cases, current collaborators guiding the newcomers is crucial. Face-to-face human guidance is invaluable since it is a two-way dialog. Doc-

Table 1: Comparison of four different experiences

| | **Experian** | **UCR** | **Rutgers** | **Intel** |
|---|---|---|---|---|
| **Worked for** | 1 year | 4 years | 9 months | 1 year |
| **Size of product** | 1M+ loc | small products | ∼5k loc | ∼5k loc |
| **Work location** | In-Office | Remote | Remote | In-Office |
| **Team size** | 7 developers | 5 developers | 2 developers | 3 developers |
| **Number of sites** | 3 sites | 1 sites | 1 sites | 2 sites |
| **Main language** | Java | Java | Php | C# |
| **Part/Full Time** | Full | Part | Part | Full |

umentation, although valuable, is a mechanism where information travels in just one direction and usually lacks detail [5].

# 3 Work Context

Between 2010 and 2014, I formally worked for four companies: Experian Credit Score Reporting Services (Experian), The University of Costa Rica (UCR), Rutgers University - The State University of New Jersey (Rutgers), and Intel Corporation (Intel). All of these companies have a different set of working environments and transmitted knowledge in different ways. Table 1 shows a summary of these experiences, including the length of time I worked for the company, the approximated size of the product I worked on, whether the work was remote or in-office, the size of my team, the number of sites the team was distributed over, the main programming language, and whether it was a part- or full-time position.

## 3.1 Experian - Credit Score Reporting Services

In the Fall of 2013, I was offered a position at a new branch of Experian - Credit Score Reporting Services, as a Software Developer II in the site located in Heredia, Costa Rica. Experian is a well-known company worldwide, with over 16,000 employees. Their business encompasses credit reporting, but also other products such as decision analytics, marketing assistance and identity theft. I learned that me and the other newcomers would be working mainly with two other branches: Costa Mesa, California, and Santiago, Chile.

The branch in Costa Mesa was well established, with some employees

having worked for over 20 years in the company. The Costa Rica branch (which I had just joined) was the latest addition to this growing community. I was the third employee in the Identity Theft field; the other two employees were hired a couple of weeks before me. When I left the company a year later, the Identity Theft area had over 25 employees and it kept growing.

I worked with a couple of different groups, each building a distinct product. In this sense, Experian was similar to how research is produced in academia: A person works with several groups composed of collaborators with different backgrounds and diverse levels of expertise.

The main project I worked on was a tool for identity theft prevention to increase the security of end users. When end users want to check their credit score, they go to the Experian website and provide authentication information. If the information provided is below a certain threshold of accuracy, our tool is invoked. We would collect information from other companies about the customer, and ask questions about personal information that only this particular individual should know( e.g.: their eye color as reported in their driver's license).

I worked on this project for a year. Throughout that year, the process of understanding this massive code base was an uphill experience for me, in part because the code was rarely documented and there were no accessible mentors to ask for guidance. Similar to me, new employees where being constantly hired for the new Costa Rican branch, and they had to go through a similar learning process. My previously mentioned experience falls into a large umbrella of challenges faced by companies when dealing with knowledge transfer. From my experience, the challenges the enterprise faced included the following:

- There is *institutional knowledge* that needs to be transferred to the new branch. There is no single expert with the entirety of the necessary knowledge to be transferred. Instead, knowledge is spread out among a large number of different collaborators. Enough of this knowledge needs to be transmitted from the employees in the older branches to the employees in the new branch for them to be able to perform their work adequately. In my experience, if I wanted to understand a particularly difficult section of code, I would ask a developer working on that code. This developer could tell me specifics about this module in particular, but would not know for example, where the module is being called or where do the parameters come from.

- The differences in pace and knowledge between the team in the well-established branch and the new branch. In the former, business knowledge is internalized and well understood by the employees; as opposed to the employees of the new branch who just started working for the company. By contrast, in the old branch the employees usually have limited knowledge of new tools, frameworks or technologies overall. In the new branch, the opposite was common: a team of young developers full of knowledge regarding the latest technologies and used to working at a very high pace, but without the business knowledge for this particular company.

- The fact that the core business logic product, the concept of a "credit score", was not commonly used or known in the cultural context of the new branch. In Costa Rica, credit scores are rarely used. This is by contrast with the United States, where customers' credit scores are checked if they want a cell phone plan, rent an apartment or open a bank account. In Costa Rica, credit scores are used mostly for long term loans such as mortgages, and companies do not require your authorization to check it. Therefore, even if your credit score is checked, you would not know about it, which makes the concept of the "credit score" not well-known or fully understood in the new branch's local culture.

- Resistance to change by the employees in the old branch whose voice has more weight because of seniority. This is contrasted with the employees in the new branch, who have new ideas that can be implemented in the company, but also lack experience and knowledge regarding how the company works. I noticed this the most when the employees of the new branch suggested the use of Jenkins to track development progress. Even though we got a thumbs up to start using it and we did, the more senior developers refused to, because they did not want to spend time learning to use a new tool.

- Difficulties introduced by location disparities such as geographical and time zone differences. These lead to challenges in the development process, specially difficulties with communication and synchronization efforts. This was more noticeable in the mornings. We started working at 7am Costa Rican time, while the team in Costa Mesa usually started working at 11am in Costa Rican time (9am California time).

This usually meant that if we needed clarification or any kind of communication, we needed to wait several hours every day to be able to interact with the more experienced team.

## 3.2 Past Experiences

It is particularly interesting to compare to my other past experiences which map to more traditional forms of knowledge transfer, because of the diversity and lessons learned they contribute to my understanding of how knowledge is transmitted.

### 3.2.1 University of Costa Rica

At The University of Costa Rica, I worked as a Web Developer and Webmaster. The institutional knowledge in this case was minimal. I worked for a research center, with a small team of developers. We constantly received requests to build small products or implement changes to existing products. These tasks usually took between one and six months to build. The previous Webmaster was a single person who knew the high level details for most of the products. He possessed, to a large extent, all the knowledge necessary to perform properly as a Webmaster in this context.

In this case, the knowledge transfer experience I had was approximately three hours with the previous Webmaster. He very kindly indicated to me the knowledge he thought was necessary for me to perform well, while I asked any questions I could think of. The biggest limitation in this experience was the time constraint: I had only one meeting with this person.

Ultimately, three hours of training was not quite enough for a position that I held for four years. It quickly became a very challenging experience for me, largely because I had to learn most of the technical knowledge and business logic on my own. This presented the steepest learning curve of all my experiences. The majority of the knowledge I obtained was not provided by anyone within the company, but rather via third party sources or by my own experimentation. For example, we used several different frameworks and plug-ins. I had to look for and read the documentation of these frameworks and plug-ins to understand how they worked. When I couldn't find documentation to read, I relied on other learning mechanisms such as modifying the source code and re-running it to compare its behavior before and after the changes where performed.

Unlike my other experiences this company had a very weak knowledge transfer mechanism. This resulted in considerable time spent by the newcomers looking for knowledge that was not passed down to them. The newcomers took much longer to be able to obtain knowledge that was already present in previous developers, and could have been learned in a more straightforward way if the knowledge was properly handed down to them in the form of documentation or face-to-face communication.

### 3.2.2 Rutgers University

My 9-month experience as a Web Developer in Rutgers University is interesting to contrast with the others because the knowledge transfer was written, not verbal. The source code was really well documented, but the person who had written the comments had already left the position. Having written documentation was advantageous because I could access it where/whenever it was most convenient for me. But it does not have the dialog that face-to-face mechanisms provide, which is particularly helpful when I had doubts not covered by the documentation.

When I applied for this position, I was first interviewed by the director of the department, a non-technical person. He broadly explained to me that I was going to perform several changes to a program used by the department for record keeping, staff training, etc. Once I started the job, the director gave me the password to the server and told me to familiarize myself with the program (approximately 5,000 LOC PHP program). At this point I had to go through the program understanding what it was supposed to do, reading the comments, running it in different ways, making small changes to the functionality and re-running it to confirm that I was properly understanding the functionality. After a couple of weeks of doing this I was able to understand generally how the program worked and how to perform the changes I was asked to implement.

### 3.2.3 Intel

Finally, I worked at Intel Corporation, where I was hired as a Software Developer in February of 2012. I joined a group of 3 people (myself included): two developers (located in Costa Rica) and one project manager (located in Arizona). Both branches had existed for over 20 years and the working culture had been well assimilated and internalized. This helped soften my

immersion into the work environment since I had to only adapt to the already existing working culture(as opposed to Experian, where we were creating a culture from scratch).

One important factor that made my transition more pleasant regarding knowledge transfer was that the other developer I was working with was knowledgeable. He knew the technologies and processes being used in this team very well. He was also highly available. We worked in the same cubicle and when I needed further explanation of any given task, his proximity and willingness to help softened my learning curve making me learn much faster. Also, since we were regular co-workers, knowledge transfer did not have a time limit (unlike my experience at the University of Costa Rica), so knowledge could flow in my direction very naturally in an ad hoc basis.

# 4 Reflection

In my experience, a hybrid approach using both face-to-face and documentation is the best and fastest way to help newcomers to adapt and understand a new landscape. In the experiences where I had a good balance between documentation and face-to-face time (such as at Intel), I was able to learn much faster and assimilate the code than I could when I had only documentation (Rutgers) or very limited face-to-face interactions with the mentors (Experian and UCR).

Previous studies have focused on the difference between explicit knowledge and tacit knowledge (learning *about* something versus learning *to do* something) [4, 3]. Both types of knowledge are important from a software engineering perspective. I focus on the differences between knowledge acquired through one-way communication (e.g. code documentation) versus two-way communication (e.g. face-to-face knowledge transfer). Documentation (one-way communication) is a common practice in software engineering, and mentors' help (two-way communication) usually involves senior developers, whose time is valuable.

I have also validated with my personal experience key insights from previous literature. For example, I have observed that early experimentation (UCR and Rutgers) and progress validation (Intel) are key features in the learning process of software projects [5]. I have also witnessed the advantages of learning from an active developer with social skills [10] and geographical proximity [11] (Intel and Experian).

## 4.1 One-way Communication

I have observed that proper internal documentation is the most important of all the forms of documentation for new developers to understand the purpose of the code they will be working with. This includes meaningful names for classes, methods and variables, clean code, and developer written comments.

The importance of this particular kind of documentation comes from the fact that it is immediate. Newcomers go through the code to try to understand it by making sense of the developer written comments and class, method, and variable names. They go through a cognitive matching process between the expected behavior of the program and the code. If such comments, functionality and component names match the developers' mental model of the program expected behavior, then the knowledge transfer will be immediate and effective.

Unfortunately, in my experience, developer written comments are very rarely updated, which results in comments that do not match what the code is currently doing. Throughout all my experiences I saw outdated comments on a daily basis, though less frequently when using software meant to be highly reproducible(e.g., frameworks). In all other cases, where understandability was not a priority, outdated comments were a common problem.

This leads to newcomers not trusting the comments. Outdated documentation is in general a well known problem in software maintenance [7] and this is a scenario where we can see in a very clear way the impact of this problem, since it interferes with the knowledge transfer process and the newcomer learning experience.

## 4.2 Two-way Communication

In my experience, two-way communication between newcomers and mentors represents a much smaller component of the knowledge transfer process. One of the main reasons behind this is that senior developers' time is a highly valued asset which must be used for high priority tasks, and often knowledge transfer to newcomers does not qualify as such.

In my experience, this kind of knowledge transfer is less necessary when the documentation in the project is of high quality. As documentation quality decreases, the need for a mentor increases. This can be described as an inversely proportional relationship between the quality of the documentation in the project and the time needed to be spent by both the mentor and the

newcomer in face-to-face communication.

In my experience at Rutgers, for example, the internal documentation of the project was of high quality. Therefore the need for a two-way documentation was lesser than my experience at Experian, where the documentation was of low quality and I needed much more mentorship.

## 4.3 Conclusion

From my experience, the most important of the different kinds of documentation, is the internal documentation which encompasses names of classes, variables, methods, and developer written comments. The importance of this comes from the immediateness of this kind of documentation to satisfy doubts the newcomers may have. Anecdotically, this is the most commonly used source of information by newcomers. Two-way communication is needed in a inversely proportional manner to the quality of said documentation.

Some high level factors that can highly improve the learning experience of newcomers when joining a new software project are:

- The presence of both documentation and an expert, with emphasis in the former
- Knowledgeable and available mentor(s)
- Include newcomers into already existing teams with an already defined work culture

I can derive from this information that applying best practices from code maintenance literature such as using meaningful names for classes, variables, and methods in the software being built, and constantly updating the developer written comments is optimal for the knowledge transfer process in newcomers. Therefore making the need for two-way communication between newcomers and mentors minimal. This is particularly advantageous in the context of knowledge transfer when opening a new branch since it maximizes the volume of knowledge gained by the overwhelming number of newcomers in the new branch, and minimizes the need for mentors, which are particularly scarce in this scenario.

# References


[1] F. P. Brooks Jr. *The Mythical Man-Month Essays on Software Engineering*. 1995.

[2] E. Ceruttia, G. DellAricciaa, and M. S. M. Peria. How banks go abroad: Branches or subsidiaries? In *Journal of Banking and Finance*, volume 31, 2007.

[3] E. Civi. Knowledge management as a competitive asset: a review. In *Marketing Intelligence & Planning*, volume 18, pages 166–174, 2000.

[4] S. Cook and J. Brown. Bridging epistemologies: the generative dance between organizational knowledge and organizational knowing. In *Organization Science*, 10, pages 81–400, 1999.

[5] B. Dagenais, H. Ossher, R. K. E. Bellamy, M. P. Robillard, and J. P. de Vries. Moving into a new software project landscape. In *International Conference on Software Engineering*, ICSE '10, 2010.

[6] F. Ibrahim and V. Reid. What is the value of knowledge management practices?, 2009.

[7] T. C. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: the state of the practice. In *IEEE Software*, volume 20, pages 35 – 39, 2003.

[8] M. Monperrus, M. Eichberg, E. Tekes, and M. Mezini. What should developers be aware of? an empirical study on the directives of api documentation. In *doi:10.1007/s10664-011-9186-4*, 2011.

[9] M. N. Razavi and L. Iverson. A grounded theory of information sharing behavior in a personal learning space. In *Computer Supported Cooperative Work*, CSCW'06, 2006.

[10] I. Steinmacher, I. S. Wiese, and M. A. Gerosa. Recommending mentors to software project newcomers. In *International Workshop on Recommendation Systems for Software*, RSSE 2012, 2012.

[11] E. Whitworth. Agile experience: Communication and collaboration in agile software development teams, 2006.

[12] K. M. Wiig. Knowledge management: Where did it come from and where will it go? In *Expert Systems With Applications*, volume 13 of *1*, pages 1–14, 1997.