The Effects of Modularity on Effective Communication and Collaboration

Full Paper

Chintan Amrit
IEBIS Department
University of Twente
The Netherlands
c.amrit@utwente.nl

Elody Hutten
APG Asset Management
Amsterdam
The Netherlands
elody.hutten@gmail.com

ABSTRACT

In this article we explore the consequences associated with a lack of coordination between the requirements engineering process and the development process. We conduct a detailed case study of an ICT department of a large European bank that develops software using the agile software development method. Our current study reveals that the application of a modular organizational design in a dynamic agile environment has a negative effect on the communication and coordination between members of different modules. More specifically, the modular design creates both a semantic and a pragmatic boundary among members of different modules, which is primarily caused by the fact that modules have differentiated tasks and often misaligned interests.

CCS CONCEPTS

•Software and its engineering~Agile software development • Software and its engineering~Programming teams • Software and its engineering~Agile software development

KEYWORDS

Agile Software Development, Modular organization, knowledge sharing boundaries

1 INTRODUCTION

Conway's law states that "organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations" [1]. This organization pattern [2, 3] implies that the interface structure of an information system will mirror the social structure of the organization that designed it [4-6]. The importance of coordination between the requirement engineering process and the development process has also been demonstrated [7]. Frank and Hartel (2009) [8] conducted a study with respect to the application of a modular organizational design within an agile software development context and found that a reintegration of different modules resulted in an increase in both team morale and team performance. These findings could indicate that splitting a tightly coupled system (e.g. a development team) into autonomous modules within an agile software development environment reduces the performance of such a system. An explanation for these observations could be the fact that modules have their own goals which might cause two different modules to become highly differentiated. This is clearly evident in the study by Frank and Hartel (2009) [8, 9], one of the modules was responsible for writing user stories while the other module was responsible for actually developing the software. A consequence of these differentiated roles could be that the development of shared mental models, which is the overlapping of the cognitive representation of the external reality between team members [9], is inhibited, due to a semantic boundary between the modules, interpretive differences despite a common lexicon [10]. Another consequence of the differentiated roles modules could have, is that it creates a pragmatic boundary, which could inhibit effective communication. Finally, the separation of a tightly coupled system into different modules could result in a decrease of relational capital between members of different modules. These hypotheses suggest that applying a modular design in an agile software development context might lead to a decrease in coordination between the different modules which, as reasoned above, is crucial in software development and especially in agile software development. It is therefore questionable whether a modular organization design can be applied effectively within an agile software development context. This leads to the following research question: *Can a modular organizational design be applied in an agile software development context?*The current study will investigate the role of a modular organizational design with respect to its effect on communication and coordination between modules composed from a tightly coupled system within an agile software development context.

SIGMIS-CPR '17, June 21-23, 2017, Bangalore, India

ACM ISBN 978-1-4503-5037-2/17/06…$15.00
http://dx.doi.org/10.1145/3084381.3084413

2 Literature Background

The framework of Carlile (2004) [10] provides an understanding of communication and boundaries affecting effective communication. Carlile (2004) constructed a framework regarding knowledge sharing within organizations. He proposed

three boundaries of knowledge sharing with increasing complexity due to increasing novelty, specialized (domain-specific) knowledge and dependency: syntactic, semantic and pragmatic with the corresponding capabilities: transfer, translation and transformation see figure 1. The first boundary, the syntactic knowledge boundary, is concerned with the lack of a common lexicon, which prevents knowledge from being processed across a (functional) boundary. A shared, stable syntax could serve as a boundary object and enable the transfer of knowledge (boundary spanning). The second boundary is the semantic boundary which is concerned with interpretive differences despite a common lexicon that decreases effective collaboration and coordination. It is necessary to consider tacit, context-specific knowledge in order to be able to span the semantic boundary and really understand the meaning of knowledge that is being transferred (called translation). Purpose of semantic boundary spanning is, thus, the development of a shared meaning. The final boundary is the pragmatic boundary which refers to conflicts that arise due to consequential interaction in the presence of conflicting interests. The purpose of pragmatic boundary spanning is to achieve a common interest.

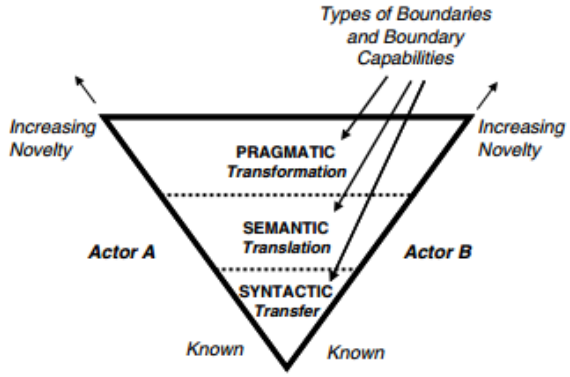

Figure 1, Integrated framework regarding knowledge sharing [10]. There are three levels of boundaries and corresponding required capabilities.

A study by Hsu, Chu, Lin, And Lo (2014) [11] investigated the integrated framework regarding knowledge sharing by Carlile (2004) [10] in an agile software development context. More specifically, they tested an extension of the model which included three aspects of intellectual capital and their influence on effective boundary spanning and the influence of effective boundary spanning on information system performance

Ernst (2006) [12] conducted a study with respect to the limits of modularity. He hypothesizes that modularity has been taken too far and that the limits to modularity are not taken into account appropriately. The conclusion that he draws from his research in the chip industry is that inter-firm collaboration requires more coordination through corporate management when codification does not reduce the complexity.

The application of a modular design in software development in an agile environment has been observed in a case study by [7]. Besides studying the consequences of lacking customer involvement, they also report on how organizations dealt with this issue. One of the strategies they found was the use of a definition of READY. This means that the requirements provided by the customer (or representatives or surrogates for that matter) need to conform to a certain standard (the definition of READY) before the developers are prepared to work on it [7].

Frank and Hartel (2009) [8] conducted a study in a company that used a modular organizational design by creating a requirement engineering model (READY) and a development module (DONE). Originally, separate teams were responsible for constructing user stories (READY) and development teams (DONE) who were responsible for building the actual software. Frank and Hartel (2009) concluded that the increased collaboration between the READY and DONE part increased team morale and more predictable results while maintaining a constant velocity. This study, thus, questions the application of a very modular design in an agile software development context [8].

Currently, a new conceptual framework with respect to software development is emerging: DevOps. DevOps extends agile methodologies and principles outside of the field of development in order to integrate two departments: development and operations[13]. DevOps clearly views the different aspects of an information system as interdependent which is an indication that these are not loosely coupled systems. The fact that the emerging conceptual framework in IS research, DevOps, advocates more integration and collaboration could be seen as another indication that a modular organizational design might be less appropriate in an agile software development environment.

As mentioned above, a modular organization design can be inappropriate in an Agile development environment, as parts of an organization are not loosely coupled systems with weak interdependency. Moreover, customer involvement is highly important in requirements engineering, which implies that the READY and DONE part are highly dependent, and thus tightly coupled [7]. So, creating modules out of a tightly coupled system can lead to a decrease in the effectiveness of the modularity. So, from an organizational design perspective, the results obtained by Frank and Hartel (2009) can be explained by the fact that modules are created from a tightly coupled system (scrum team) which results in a decrease in coordination and, consequently, a decrease in performance [8].

Our research question will be answered using a case study approach within a program concerned with agile software development. Data were primarily collected using qualitative interviews.

3. Case Study

Our case study setting is at a bank in Europe, referred to as 'European bank' in this paper. The ICT group of European bank have adopted agile software development practices based on the scaled agile framework (SAFe) [14]. Just as in many agile projects, this case followed a modular organizational design [8]. Two modules were created from the scrum teams, a READY part which is responsible for getting user stories "READY" and a DONE part which is responsible for getting the user stories "DONE". One of the core reasons for implementing the agile model was to remove project management from the development part. This was achieved by dividing the original scrum teams in both a READY and a DONE team whereby the project manager was included in the READY team. It was thought that, by decoupling the DONE teams from a certain project and thus single project manager, teams would become more stable since they do not have to be abrogated after a project has finished, but the DONE teams can now be assigned to another project, or used for different projects at the same time. Another benefit was that the DONE teams would now be coordinated by one person who would become responsible for the continuity of these teams and, thus, the HR. It was therefore hypothesized that the program would be able to add more value by being able to give priority to features and stories at program level instead of giving priority to user stories within projects. Another consequence of this arrangement was supposed to be improved scalability, the teams could become more able to respond to changes. When a higher demand for developers occurs, new teams could be created and when demand drops, teams could be send home.

The design of the Agile model occurred using working groups, consisting of all the people that were interested. In total, there were six themes and six working groups. These themes were: communication, process decisions, governance, team/aligned functions, environments and scrum of scrums. Each working group thus worked on how the new model should address the issues related to their theme. After approximately one year, the design of the Agile model was finished and the new agile process model was made definite and implemented. The actual implementation consisted of a presentation to inform the people of the program about the new model.

The scrum teams (also called "DONE teams") were responsible for building the software and consists of a scrum master, product owner and developers and tester. There were approximately 18 DONE teams. All members of the DONE teams were externals and both European and Indian nationalities were present in (some) of the teams, although no DONE team were entirely made up of people from India. Most of the DONE teams were collocated, with the exception of teams including Indian developers or tester since these team members were often located in India. Contact with these globally distributed team members was established via telephone, email, chat and videoconferencing. In principle, the Indian team members participated in all scrum rituals.

The READY teams were responsible for transforming business requirements (from different projects) into user stories that could be built by DONE teams. These teams consisted of a product manager, project manager, application engineer, test manager, interaction designer and a business analyst. These READY teams were organized among six themes: financial insight, cross-channel functionalities (one theme is concerned with banking related functionalities and the other theme with the remaining functionalities), content interaction and design, cross-channel marketing, integration of the client's world and cross-channel contact. The exact composition of the READY teams was not rigid and varied per theme.

The current study uses semi-structured interviews using the framework provided by Myers and Newman (2007) to conduct semi-structured or unstructured interviews in an information systems context. We use the interview method to gain a better and more in depth understanding of the effect of a modular organizational design on the knowledge sharing in an Agile development enviroment. We engaged both a "maximum variation" and a "snowball" sampling strategy in line with the typologies provided by Miles and Huberman (1994). The aim of a maximum variation strategy is to compose a sample that is heterogeneous. The logic behind this is that the results of the extremes will aggregate to a result that is representative for the entire population. After each interview, the researcher asked the interviewee who would also be interesting to interview, hence the snowballing. These strategies and the quota of at least one informant per function resulted in a total sample of 21 different informants. More detailed information with respect to the composition of the sample can be found in Appendix A (table 1).

All interviews were face-to-face and took place in a familiar location for the interviewee. With permission of the interviewee, the entire interview was recorded with a laptop and no notes were made in order to contribute to the feeling of a natural conversation. Afterwards, transcripts were made of the interviews. The script used during the interview can be found in Appendix B.

The data analysis process occurred according to Miles and Huberman (1994)[15]. All codes were assigned using the software "Atlas TI". The coding process resulted in 58 unique codes (see Appendix C). Coding occurred by assigning anything from words to short sentences to meaningful pieces of transcript. Despite the fact that the sample size was predominantly determined by quota sampling, it was checked whether saturation of the codes had occurred after all interviews were conducted. The saturation process is illustrated by Figure 2.

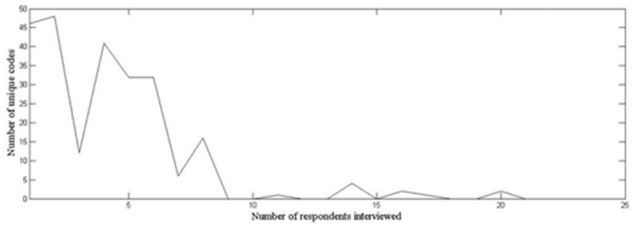

Figure 2. The saturation process.

Figure 2 illustrates that saturation of the codes occurred relatively early in the process. The dip that can be observed from the second respondent until the fifth can be explained by the fact that these interviews were conducted with line management. Their interviews contained relatively large amounts of context information because they are further removed from the actual process in comparison to members of both modules and the data they provided was more concerned with how they expected the agile process to work, compared to how it actually worked in practice.

After all transcripts had been coded, all codes thematically related were clustered and, if necessary, refined (Figure 3). From these clusters of codes, categories with respect to the research question were constructed.

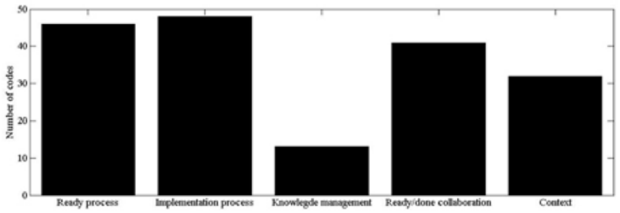

Figure 3 Number of codes per category. Categories from left to right: ready process, implementation process, knowledge management, ready/done collaboration and context.

Figure 3 illustrates that the READY process, implementation process and READY/DONE collaboration were mentioned approximately equally often across all the interviews. Knowledge management was discussed less often which could indicate that knowledge management is less important and/or less of an issue within the process.

4. Results

This section contains the results obtained by the current study. As part of the results section, quotations were added. Since the interviews were conducted in a western European language, the quotes were translated. The results section will explain what the interviews revealed the consequences of the modular design.

4.1 The semantic boundary

During the interview, people were asked about the implementation of the modular design and the consequences thereof. One of the things that popped up during the interviews is that people perceived the implementation process of the new modular design that was chosen to be insufficient. The implementation consisted of a presentation that was given to all members of the program. During the presentation, the philosophy of the new modular agile model was discussed, as well as the benefits it was supposed to bring in terms of flexibility at the program level. However, the presentation did not explain the impact of the model beyond the operational level, in terms of changes in the different roles (like the business analyst), the requirement engineering process and the development process. One interviewee indicated that they had expected a more active implementation containing change management. The perception that the presentation was not sufficient to cover the entire implementation process is illustrated by quote 1.

Q1: *"We were given a presentation containing the general idea of what they were aiming for and I've got the feeling that it has not sunk in, at least not sufficiently. You have to really implement it."* (P07 – Test Manager)

An example of a concept that does not have the same meaning across the entire department but is used by all the teams of both modules is a "feature". Customer demand is captured in a business case which describes broadly what new functionality should be able on the company's website for example. This business case is split up in features and from these features, user stories are created. The three concepts are distinguishable by the amount of time needed to realize them whereby the business case requires the most time and the user stories the least. This broad distinction is known across the department but exact definitions of the different concepts are not the same. This is illustrated by quote 2 by a Business Analyst:

Q2: *"How can you, if you cannot even uniformly determine the weight of a feature, compare features across teams? (..) It becomes very difficult to exchange a feature from team one to team two when team one and team two think differently."* (PO8 – Business Analyst)

This quote illustrates the semantic boundary between members of different teams, apparently the concepts are known across the entire department but their meaning is different for teams and thus also varies between both modules. These results are also in line with the model as proposed by [11], who concluded that a common understanding positively affected effective communication.

An Application Engineer mentioned (quote 3) that the semantic boundary has to be spanned using very elaborate specifications in the modular organizational design. These specifications are needed in order to create a common understanding and to enable DONE teams to build software.

Q3: *"The way we have it [modular organizational design] is that a DONE team has to be able to build software without the context that we [READY] have. Practice shows that when you conduct the*

process like this [modular organizational design] is that specifications need to be very elaborate or a DONE team will not be able to estimate the work and execute it." (PO 20 – Application Engineer)

Apparently, the implementation of a modular design in the case that was investigated has resulted in a lack of shared mental models between the teams. The meaning attached to different concepts as well as representations of the development process are not necessarily similar across all the teams. In conclusion, it appears that the modular design had created a semantic boundary thereby causing a lack of shared mental models across the teams. The effect of this lack of shared mental models caused by the semantic boundary between teams, had caused a decrease in the effective communication and coordination between teams. The interdependencies between the teams of the READY and DONE module created the necessity to coordinate and communicate between them. However, effective communication and coordination was inhibited by the lack of shared mental models since a shared understanding has to be reached before any coordination can occur.

4.2 The pragmatic boundary

Another topic discussed during the interview was the split of responsibility between the READY and DONE teams. Although both modules worked on the same product, this common goal/responsibility was not experienced in practice. When, for example, a new function for the mobile application had to be developed the READY module solely felt responsible for writing the user stories and after they had finished, their responsibility for the application ended. The same could be applied to the DONE module: they are merely responsible for actually developing the application and they did not feel responsible for the user stories.

Q4: *"Now, they [READY] do not have the responsibility to deliver something. That responsibility now lies solely with us [done] and they merely prepare at the moment (…). Maybe that should be more aligned so that the people that create the specifications also feel responsible for the delivery."* (PO12 – Developer)

This modular design and the relatively strict division of the responsibilities of both modules was also perceived to be very "waterfall like" by some of the participants in line with the observations of Frank and Hartel (2009)[8]. With this, participants were most often referring to the tendency of both parties to "throw it over the wall", meaning that READY "throws" their user stories over the wall and that DONE has to figure out what they are supposed to build but the other way around also occurs, DONE throws "not READY" user stories back over the wall and READY has to fix it. This effect is illustrated by a scrum master in quote 5.

Q5: *"READY and DONE is a waterfall. You have got the preparatory work [writing user stories] and it is just thrown over the wall and it has to be done."* (PO 9 – Scrum master)

This finding indicates a pragmatic boundary that is negatively affecting collaboration. Like mentioned before, Carlile (2004) defines the pragmatic boundary as conflicts that arise due to consequential interaction in the presence of conflicting interests meaning that conflicts arise when their goals regarding knowledge delivery contradict [10]. Another factor that was used to explain the pragmatic boundary was the fact that READY and DONE were not working in parallel but sequential. READY first wrote user stories and after they are finished, DONE would build them. During the development process, however, READY was already working on something else which could decrease their interest in the user story that DONE was working on.

It appears that both the semantic boundary and the pragmatic boundary indeed cause a decrease in effective communication and that the semantic boundaries occurs both within and between modules and that the pragmatic boundary solely occurs between modules. No evidence was found with respect to the occurrence of a syntactic boundary based on the results of the current study.

4.3 Intellectual capital: the state of relational capital

Besides the "throw it over the wall" effect, the modular design also had an impact on the relationship across members between different modules. The results reveal that an "us and them" feeling could be observed between the modules meaning that members from the READY module felt little affiliation with members from the DONE module and vice versa. This lack of affiliation has negatively influenced the relationship between members of both modules. Quote 6 by a business analyst illustrates what the effect of the modular design is on the relationship between the ready and done modules.

Q6: *"I do not have a relationship with them [DONE]. I'm so far removed from them; I just have to deliver user stories".* (PO8 – Business Analyst)

This view is similar to another quote by a member of a DONE team that also stresses the consequences of the decreased relationship between both parts (quote 7).

Q7: *"I do not consider them [READY] to be part of us [DONE]. The way you look at them and approach them does differs."* (PO12 – Developer)

The fact that the quality of their relationship has decreased due to the fact that they have become members of different modules has affected the way they approach and interact with each other. One of the consequences mentioned during one of the interviews is reluctance from both parties to initiate face-to-face contact. This face-to-face contact is often replaced by emails which delays the response and decreases the richness of information thereby causing inefficiencies. This effect was illustrated by a Developer (quote 8) who indeed felt that not knowing each other

and being located at different floors inhibits the initiation of face-to-face contact. As a consequence, this contact was initiated using emails whereby important nuances of the message are lost.

Q8: *"At first, you do not know each other. They say that it does not matter whether you have a development team on the second floor and you have to take the stairs (…). But when you do not know each other, you would rather send an email but then you will miss the nuance."* (PO 14 – Developer)

The fact that a decrease in the quality of the relationship between both modules has led to a decrease in effective communication between the modules which, in turn, has caused ineffective coordination is in line with the results obtained by [11] who found that a decrease in relational capital negatively affect communicational effectiveness which results in a decrease in project efficiency.

These results further confirm the theory that the strictly separate responsibilities of both modules causes a pragmatic boundary and a decrease in effective communication by proving that this decrease in effective communication does not happen when the responsibilities are not strictly separated. The same reasoning applies to the theory that a decrease in relational capital results in a decrease in effective communication since the current study found that effective communication remained when the quality of the relationship between both modules remained intact. This finding is also in line with the literature on DevOps which advocates a culture of open communication and trust in order to enhance collaboration.

5. Conclusion and discussion

The aim of the current study is to investigate the degree to which a modular organizational design can be applied effectively in an agile software development context. More specifically, the current study tried to explain the limits of a modular design in an environment where technologies change fast and unpredictably as found by Ernst (2006) [12]. In order to gain more understanding with respect to this finding, the effects of applying a modular design in an agile software development context on effective communication was researched.

One of the results was that the modular design caused a semantic boundary between members of different modules which indicates a lack of shared understanding. It appeared that this semantic boundary was caused primarily by the different goals assigned to each module. Due to the fact that each module had its own goal(s), each module had its own processes and methodologies which were not known by heart by members outside of the specific module. So, the different concepts and processes applied by different modules causes a semantic boundary between modules. Another effect of the modular design is that, although some concepts and processes are used across all modules, the meaning attached to these concepts and processes differs across modules. This means that the modules have a common lexicon but do not necessarily have a shared meaning. This lack of shared understanding implies that the semantic boundary caused by the modular design also inhibits the development of shared mental models between modules. The effect of the lack of shared mental models caused by the semantic boundary is that a decrease of effective communication between members of different modules can be observed. This result is in line with the studies by [10, 11, 16]. The differentiated roles of the different modules could explain the lack of shared mental model development between members of different modules.

Another result of the current study is that the modular design resulted in a pragmatic boundary between the modules. More specifically, the different goals of the different modules resulted in misaligned interests thereby decreasing the incentive of members of different modules to communicate with each other. This effect appears to be very similar to the "throw-it-over-the-wall" effect as observed by [17]. After each module had completed their task, they "threw their work over the wall" to the other module who had to figure out how to deal with it. For example, the READY module was responsible for writing user stories which the DONE module had to build. After the READY module had completed their task, they presented their user story to the DONE module which had to deal with it relatively autonomously. Due to the fact that the formal responsibility of the READY module ends after the user story is completed, they lack an incentive to help and guide the DONE module during software development. Therefore, in this case, a modular design causes a pragmatic boundary between modules which inhibits effective communication between members of different modules. These results are in line with the study by [10]. Further, these findings are also in line with the literature on DevOps, which advocates aligned responsibilities among the parties involved in information systems.

Finally, the results of the current study revealed that the modular design decreases the relational capital between members of different modules. In practice this decrease in relational capital resulted in an "us and them" feeling between members of the different modules. The effect of this group thinking is that members of other modules were approached different from members of the same module. Communication between members of different modules with relatively low relationship quality were often approached either through other, better known people or via communication channels with lower quality compared to face-to-face communication like email. The effects of these coping strategies resulted in a decrease in effective communication between the members of different modules with a relatively low quality relationship. This negative effect of a decrease in relational capital on effective communication is in line with the results found by [11]. Again, this finding is in line with the literature on DevOps which stressed the importance of a culture of open communication and trust.

In conclusion, the current study has revealed that the application of a modular organizational design in a dynamic agile environment is limited due to the fact that a modular design has a negative effect on the effective communication and coordination between members of different modules. More specifically, the modular design creates both a semantic and a pragmatic boundary between members of different modules which is primarily caused by the fact that modules have differentiated tasks and often misaligned interests. Another consequence of modularity is a decrease in the quality of the relationships between members of different modules which are thought to be caused by group thinking. The effects of the decrease in effective communication and coordination between members of different modules are more severe for modules that are highly interconnected.

Acknowledgements

The authors would like to acknowledge the contribution of Dr. Michel Ehrenhard for his guidance of this research project.

REFERENCES

[1] Conway, M. E. How do committees invent. *Datamation*, 14, 4 (1968), 28-31.
[2] Amrit, C. and Van Hillegersberg, J. Detecting coordination problems in collaborative software development environments. *Information Systems Management*, 25, 1 (2008), 57-70.
[3] Coplien, J., O. and Harrison, N., B. *Organizational Patterns of Agile Software Development*. Prentice-Hall, Upper Saddle River, NJ, USA, 2004.
[4] Amrit, C. *Improving coordination in software development through social and technical network analysis*. University of Twente, 2008.
[5] Herbsleb, J. D. and Grinter, R. E. *Splitting the organization and integrating the code: Conway's law revisited*. ACM, City, 1999.
[6] Amrit, C., van Hillegersberg, J. and Kumar, K. Identifying coordination problems in software development: finding mismatches between software and project team structures. *arXiv preprint arXiv:1201.4142* (2012).
[7] Hoda, R., Noble, J. and Marshall, S. The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology*, 53, 5 (2011), 521-534.
[8] Frank, A. and Hartel, C. *Feature teams collaboratively building products from ready to done*. IEEE, City, 2009.
[9] Klimoski, R. and Mohammed, S. Team mental model: Construct or metaphor? *Journal of management*, 20, 2 (1994), 403-437.
[10] Carlile, P. R. Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization science*, 15, 5 (2004), 555-568.
[11] Hsu, J. S.-C., Chu, T.-H., Lin, T.-C. and Lo, C.-F. Coping knowledge boundaries between information system and business disciplines: An intellectual capital perspective. *Information & Management*, 51, 2 (2014), 283-295.
[12] Ernst, D. Limits to modularity: reflections on recent developments in chip design. *Industry and Innovation*, 12, 3 (2005), 303-335.
[13] Erich, F., Amrit, C. and Daneva, M. *Cooperation between information system development and operations: a literature review*. ACM, City, 2014.
[14] Laanti, M. *Characteristics and principles of scaled agile*. Springer, City, 2014.
[15] Miles, M. B. and Huberman, A. M. Qualitative data analysis: A sourcebook. *Beverly Hills: Sage Publications* (1994).
[16] Espinosa, J. A., Slaughter, S. A., Kraut, R. E. and Herbsleb, J. D. Team knowledge and coordination in geographically distributed software development. *Journal of management information systems*, 24, 1 (2007), 135-169.
[17] Al-Rawas, A. and Easterbrook, S. Communication problems in requirements engineering: a field study (1996).

Appendix A

Table 1 Overview with respect to the composition of the sample used in the current study.

Function/role	Number	Percentage (%)
Line manager	3	14,3
Project manager	1	4,8
Test manager	2	9,5
Application engineer	1	4,8
Product manager	1	4,8
Business analyst	3	14,3
Interaction designer	1	4,8
Scrum master	3	14,3
Product owner	1	4,8
Tester	1	4,8
Developer	3	14,3
Total	21	100

Appendix B

The interview protocol:

1. Opening
a) Introduction of the interviewer and the research
b) Aim of the interview and explanation of the structure
c) Emphasis of confidentiality and permission regarding recording was asked

2. General information interviewee
a) Interviewee was asked to introduce his/herself
b) The following topics had to be discussed:
- Number of years active in Rabobank
- Different functions the interviewee has been operating in
- Current function and responsibilities

3. Rabobank and agile
a) Implementation of the scrum process
Topics that were included are:
- Construction of user stories
- Relevant rituals
b) Issues related to the current implementation of the scrum process

4. Definition of both READY and DONE
a) The READY/DONE process
- Goal of the division
- Implementation of the new model
- Responsibilities of both
- Formal/informal meetings
b) Issues related to the READY/DONE division

c) Solution for these issues
- Solutions within the current process
- Solutions outside of the current process

5. Ending
a) The interviewee is asked whether he or she would like to add something to the interview
b) Check whether the interviewer knows other people that might be interesting to interview
c) Thank the interviewee for his/her time

Appendix C

The codes and their nesting displayed in relational context. The core concept displayed in the center, with high-level concepts.