Organizational design and agile software development:

The effects of modularity on effective communication and collaboration

Elody Hutten – s1009028

Masterthesis Business Administration – Information Management

1st supervisor University of Twente: dr. Chintan Amrit

2nd supervisor University of Twente: dr. Michel Ehrenhard

1. Introduction

1.1 Research Motivation

As early as in 1994, failures in IT-projects caught the attention of scholars and it was reported that 31.1% of all IT project were canceled before they are completed and 52.7% are costing 189% of the initial estimated costs (Standish Group International, 1994). The Standish study defined project failure as either a project that has been canceled or a project that does not meet its budget, delivery, and business objectives. Conversely, project success, is defined as a project that meets its budget, delivery, and business objectives. They further found that as little as 16.2% of the IT projects can be classified as successes. More recently, the results of a study on the success rates of IT projects of the Dutch government stress the legitimacy of the results by the Standish Group International (1994). Elsevier (2014) concluded that 36% of the software that is created by the projects are never implemented and 57% of the projects fail in line with the definition of Standish Group International (1994) based on a committee that was set up temporarily to investigate ICT within the Dutch Government. Of all large IT projects conducted by the government, only 7% turned out to be a success. The above mentioned failure rates result in an annual loss of 4,5 till 5 billion euro. These recent numbers indicate that IT failures are still very common and are costing companies, governments and societies a lot of money. Consequently, research regarding the factors of success and failures of IT projects are of great importance for companies, governments and individuals.

1.2 Research gap

The inherent complexity of software makes it difficult to manage and to coordinate (Brooks, 1987). A related concept is Conway's law: "organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations" (Conway, 1987). This law implies that the interface structure of an information system will mirror the social structure of the organization that designed it. Socio-technical congruence means that "organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations" (Cataldo, Hersleb & Carley, 2009). The importance of socio-technical congruence in software development has been stressed before (Amrit, 2008; Amrit, Hillegersberg & Kumar, 2012), but the importance of coordination between the requirement engineering process and the development process has also been demonstrated (Hoda, Noble and Marshall, 2011). The consequences of lacking customer involvement were: the pressure to over-commit due to fixed-bid contracts, problems in gathering and clarifying requirements, problems in

prioritizing requirements by the customer (representative) due to a lack of understanding with respect to the concept, problems in securing feedback, loss of productivity due to delayed availability of requirements for example and in the worst case: business loss. This article clearly demonstrates the consequences associated with lacking coordination between the requirements engineering process and the development process.

Frank and Hartel (2009) conducted a study with respect to the application of a modular organizational design within an agile software development context and they 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 each 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), 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 (Klimoski & Mohammed, 1994), is inhibited due to a semantic boundary between the modules, interpretive differences despite a common lexicon (Carlile, 2004). Another consequence the differentiated roles of modules could have is that is creates a pragmatic boundary, consequential interaction in the presence of conflicting interests, which could also 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, like 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:

“To what degree can a modular organizational design be applied in an agile software development context?”

In conclusion 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.

1.3 Methodology

The aforementioned research question will be answered using a case study consisting of a program within the ICT department of a Dutch bank that has implemented a modular design to restructure the composition of their development department a couple of years ago. The fact that their development teams worked in an agile manner made this an appropriate case for the current study. The case study research was conducted according to the case study methodology of Yin (2003). The current study is a qualitative study that conducted semi-structured interviews and these interviews were analyzed according to the methodology presented by Miles and Huberman (1994). Members of both modules that were present within the case study were interviewed as well as line management. Further, it was ensured that one of each of the most common roles within the department were interviewed. The data was analyzed using Atlas TI.

1.4 Results

The results of the current study revealed that the implementation of a modular design in the case that was studied 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 which seem to indicate that a modular organizational design casus a semantic boundary. Further, the semantic boundary appeared to inhibit the development of shared mental models between members of different modules. The consequence of the lack of shared mental models and the semantic boundary between modules is that effective communication and coordination between members of different modules was decreased.

Further, a pragmatic boundary was found between different modules. The differentiated roles of the modules appeared to create a conflict of interest between members of different modules which decreased effective communication between the modules.

Finally, the current study has provided evidence with respect to the negative effect of a modular organizational design on the relational capital between members of different

modules. Further, this decrease in the quality of the relationship between members of different modules also has a negative effect on the effective communication between members of different modules.

Apparently, the modular design negatively affected communication between members of different modules which appeared to negatively affect the coordination between the modules. Due to the fact that coordination is crucial within an agile software development context these results indicate that a modular organizational design might be less appropriate in a tightly coupled system like an agile software development context.

1.5 Organization of the thesis

The second chapter, the literature review, will discuss relevant studies with respect to effective communication and coordination within an agile software development context. The section will conclude by providing the research question and the corresponding hypotheses. After the literature review, the case that is investigated in the current study will be discussed as well the environment of the case study. The fourth chapter will elaborate on the methodology of the current study and in this section the qualitative approach of the case study will be discussed in more detail. Finally, the results of the current study will be presented as well as the most important conclusions. The thesis will conclude by discussion the conclusions of the current study and evaluate its validity.

2. Literature review

This section will further explain relevant research regarding inter-team communication in agile software development. The first part will explain Conway's law and requirements engineering in an agile context and the importance of customer involvement. The second part will elaborate further on organizational design, communication and boundaries that can affect effective communication.

2.1 Coordination in software development

Brooks (1987) describes some essential issues with respect to software development. Among others, he mentions changeability. Changeability refers to the tendency of software to be under pressure to change. This pressure is caused by the fact that software often embodies the functionality of a system and the functionality is something that is pressured to change the most. Further, software is something that is "in the mind" or "thought stuff" which makes it

very susceptible and easy to change. Changeability, among other factors, makes software development processes difficult to manage and to coordinate. In organizational theory, coordination has been defined as "the managing of interdependencies between activities" (Malone & Crowstone, 1944) or as "directing efforts toward achieving common and explicitly recognized goals" (Blau & Scott, 1962). Kraut and Streeter (1995) stress the importance for different people working on the same software development project to define a common goal and to share the knowledge and net their activities necessary to accomplish the predefined goal(s). Hersleb and Grinter (1999) more specifically define certain aspects that have to be addressed with respect to coordination in software development. According to them, successful coordination depends on addressing what is to be developed, how it is developed, when certain activities should take place and who is responsible for what. Malone and Crowston (1990) identify four coordination processes: the management of shared resources, producer-consumer relationship, simultaneity constraints and task-subtask relationships. Supplementary to these four core processes are three support processes: communication, decision making and the perception of common objects. The difficulty of the changeability of software on coordination can be explained by the fact that changeability causes the coordination effort to alter during the project along with the changes in the software. Changes in the coordination effort further affect the coordination activities as well as the support process which is, among others, communication. This was also stressed by Wagstrom and Hersleb (2006) who emphasize the difficulty with respect to ensuring sufficient communication and coordination in an agile and (especially) a distributed environment due to the fact that the coordination requirements are changing over time and that the agile methods rely heavily on communication. Further, Nidumolu (1995) found that coordination affected performance of software projects. More specifically, vertical coordination reduced project uncertainty and residual performance risk. Horizontal coordination led to higher level of overall performance. The aforementioned studies emphasize the importance of research with respect to coordination in an agile software development context.

A concept related to coordination is called Conway's law, which states that "organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations" (Conway, 1968). This means that the organizational ties should match the task interdependencies, this alignment is also known as socio-technical congruence or the mirroring hypothesis (Cataldo, Hersleb & Carly, 2009). A socio-technical structure clash occurs when the social-technical pattern that exist during a

software development project entails a social network within a project team that does not match the technical dependencies within the software architecture that is being developed, as defined by Amrit and Van Hillegersberg (2008). In the field of product engineering, socio-technical structure clashes has been researched by Sosa et al. (2004). They found that, although there is a strong tendency for design interactions to be aligned with team interactions, both organizational and system boundaries can cause misalignment and, thus, a socio-technical structure clash. A study by Ovaska, Rossi and Marttiin (2003) researched coordination in the context of multi-site software development. They concluded that coordination of activities was insufficient and that especially the dependencies between activities should be coordinated in order to be able to achieve a common goal. Cataldo, Wagstrom, Hersleb and Carley (2006) stress the difficulty of achieving effective coordination outside formal teams. Social and communication barriers pose important obstacles for effective coordination between members of different teams. The importance of socio-technical congruence, as Conway's law refers to, was also stressed by Amrit (2008). He focused on technical dependencies within the development process based on code dependencies. Like mentioned in the beginning of this section, one of the core coordination processes is concerned with the customer-producer relationship. The next section of the literature review will elaborate further on the customer-producer relationship and coordination.

2.2 Requirements engineering and customer involvement

This section will explain requirements engineering (in an agile software development context) and the importance of customer involvement. Requirements Engineering (RE) is "the process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed". Several studies have looked at this process by investigating the communication and coordination between the customer and the developers. Communication problems in the requirements engineering process between actual customers and developers within a software development context has been researched by Al-Rawas and Easterbrook (1996). In summary they found four problems: one-way communication channels, the "notations war" (inability of functions to talk at the same level), organizational barriers (organization inhibition of cross-team communication) and the inability to trace requirements back to their human source. Apparently communication problems also arise in the requirement engineering process which had a negative effect on the success of RE. Another study, by Ramesh, Cao and Baskerville (2010) conducted a study regarding the

organization of requirement engineering and customer involvement within an agile software development context. Six practices regarding RE were identified alongside problems associated with RE in an agile context. The six practices included face-to-face communication over written documentation, requirements engineering is conducted in an iterative manner, the requirements are prioritized, due to changing requirements planning is constant, prototyping and review meetings and acceptance tests. The corresponding challenges are: problems with costs and schedule estimation, inadequate or inappropriate architecture, neglect of non-functional requirements, customer access and participation, prioritization on a single dimension, inadequate requirements validation and a lack of documentation. Apparently, communication problems also occur in an agile development context since they found that a problem arising from agile requirements engineering practices is the lack of customer (representative) or even surrogate participation due to both geographical distribution and time constraints. Ramesh, Cao and Baskerville (2010) further argue that these problems pose a great risk to the RE process which is in line with the results obtained by Al-Rawas and Easterbrook (1996) who also conclude that communication issues decrease the success of the RE process. A study by Hoda, Noble and Marshall (2011) investigated the causes and impact of lacking customer (representative) involvement on self-organizing teams within an agile software development context. They observed six different causes of a lack of customer involvement, which are:

1. Skepticism and hype, whereby customers were skeptical regarding agile software development.
2. Distance also appeared to inhibit adequate customer involvement.
3. A lack of time commitment from the customer (representative).
4. Unwillingness of large customers to engage in the project as an agile customer.
5. Fixed-bid contracts, which leave little room for agility.
6. Ineffective customer representative, who didn't understand agile or is unable to provide adequate requirements and corresponding feedback.

The consequences of this lack of customer involvement are:

1. The pressure to over-commit due to fixed-bid contracts.
2. Problems in gathering and clarifying requirements. Problems in prioritizing requirements by the customer (representative) due to a lack of understanding with respect to the concept.
3. Problems in securing feedback.
4. Loss of productivity due to delayed availability of requirements for example.

5. In the worst case: business loss.

This study by Hoda, Noble and Marshall (2011) gives an impression of the (severe) consequences of lacking interaction between the development team and the customer (representative) and further underscore the importance of good collaboration between customer (representatives) and developers during the RE process which makes research with respect to factors influencing the collaboration between the RE process and the development process of great importance. In order to better understand coordination, the next two sections will describe team communication (2.3) and will consider some theories with respect to effective communication (2.4).

2.3 Team communication

A team can be described as a "a distinguishable set of two or more people who interact, dynamically, interdependently, and adaptively toward a common and valued goal/objective/mission, who have each been assigned specific roles or functions to perform" (Salas, Dickinson, Converse, & Tannenbaum, 1992). This section will investigate team communication by looking at the effect of shared mental models between members (2.3.1) and the transfer of tacit knowledge and transactive memory systems (2.3.2).

2.3.1 Shared mental models

Like mentioned before, coordination between members from different formal groups is hindered by social and communication barriers (Cataldo, Wagstrom, Herbsleb & Carley, 2006). However, they further suggest that coordination might improve due to the development of a shared mental model over time if coordination requires the involvement of the same developers. A shared mental model is the overlapping of the cognitive representation of the external reality between team members (Klimoski & Mohammed, 1994). The benefit of a shared mental model is that both communication and coordination between team members can occur through implicit coordination (Kleinman & Serfaty, 1989). Implicit coordination "occurs when a team achieves tacit coordination of tasks and communication without direct or purposeful communication of task strategies among the participants" (Rico et al., 2008). Through implicit coordination, shared mental models are able to improve team coordination (Espinosa, Kraut, Slaughter, Lerch & Hersleb, 2001). Interestingly, these shared mental models between team members, do not simply evolve over time. Levesque, Wilson and Wholey (2001) studied the development of shared mental models in teams in a software development context and they found that the mental models of the team members with respect

to their work and the expertise areas of the other team members did not become more similar simply by time passing. These results were explained by another finding: as role differentiation increased in these teams, communication decreased which resulted in a decline in shared mental models. Apparently, coordination and communication between team members is improved by an increase in the number of shared mental models between those members. Further it can be reasoned that differentiated roles of both members inhibit the emergence of shared mental models and therefore hinders coordination and communication. It would be interesting to research whether the negative effect of differentiated roles on the emergence of shared mental models and, thus, coordination and communication is also present between members of different teams. If this relationship is assumed to hold, the differentiated roles of both the development team and their customer (by proxy) could explain an occurring lack of communication and coordination between both.

2.3.2 Transactive Memory System (TMS)

A special type of a shared metal model is a transactive memory system (TMS). A transactive memory system can be defined as "the shared division of cognitive labor for encoding, storing, and retrieving information based on a collective awareness of where specialized knowledge resides in the team (Lewis & Herndon, 2011). A transactive memory system can be seen as a shared mental model between team members which is concerned specifically with the allocation of knowledge and expertise within the team. Agile software development methods rely more on the transfer of tacit knowledge compared to more plan driven methods (Nerur & Balijepally, 2007). Tacit knowledge can be defined as "knowledge that can't be articulated". Compared to explicit knowledge, tacit knowledge is difficult to share with others through both verbal and written communication (Polanyi, 1966). Ryan and O'Connor (2013) investigated the link between the presence of a transactive memory system in a team and team performance and they found that the presence of a transactive memory system was associated with team effectiveness. This study provides evidence with respect to the importance of a transactive memory system for the effectiveness of a team in an agile context. Another study related to transactive memory systems has been conducted by Liao et al. (2014). One of their findings is that the positive effect of a transactive memory system on the quality of communication is mediated by group identification. This means that a collective sense of team identification can explain the positive relationship between a transactive memory system and the quality of communication. They also found an interaction effect between professional identification (identification of a person with their profession) and group identification and

the transactive memory system. More specifically, they found that the positive relationship between group identification and transactive memory systems was more pronounced when professional identification was low. However, in teams with high professional identification, well developed transactive memory systems were found in both low and high group identification. Apparently, group identification makes the positive relationship between transactive memory systems and the quality of communication more pronounced. This could mean that, when team members actually identify themselves with the team they are part of, transactive memory systems (or, among others, collective awareness of were knowledge is allocated within a team) have a more positive effect on the quality of communication.

In conclusion, it appears that shared mental models have a positive effect on the quality of team communication (Espinosa, Kraut, Slaughter, Lerch & Hersleb, 2001). More specifically, transactive memory systems, which are shared mental models concerned with the allocation of knowledge within a team, have a positive effect on the quality of communication within teams (Liao et al., 2014). However, the development of these shared mental models (and, thus, arguably transactive memory systems) are inhibit when role differentiation increases. Further, when group members actually identify themselves with the group the positive effect of transactive memory systems on the quality of communication is enhanced. The idea that role differentiation inhibits the development of shared mental models and thus negatively affects the quality of communication among team members could also be extrapolated to team members of different teams. Arguably, if two members of different teams have differentiated roles this could inhibit the development of shared mental models (specifically transactive memory systems) between both team members thereby decreasing the quality of the communication between both team members.

2.4 Boundary spanning and agile software development

In order to obtain a deeper understanding with respect to communication and boundaries affecting effective communication, the framework of Carlile (2004) will now be introduced. Carlile (2004) constructed a framework regarding knowledge sharing within organizations. He proposes 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.

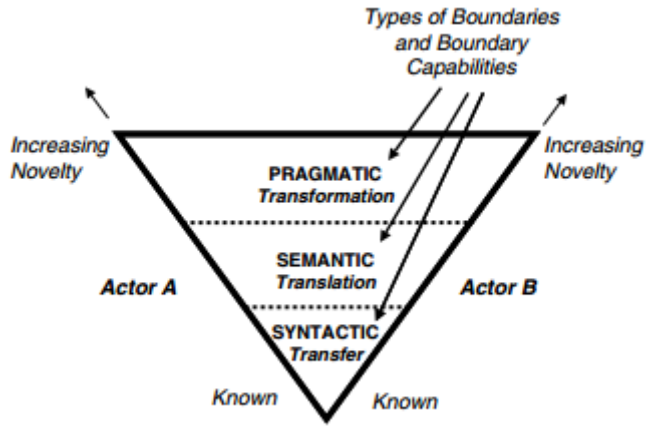


Figure 1, Integrated framework regarding knowledge sharing (Carlile, 2004). There are three levels of boundaries and corresponding required capabilities.

The first boundary, the syntactic knowledge boundary, is concerned with a lacking common lexicon. This lacking common lexicon 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 meaning that conflicts arise when their goals regarding knowledge delivery contradict. Purpose of pragmatic boundary spanning is achieving a common interest.

A study by Hsu, Chu, Lin, And Lo (2014) investigated the integrated framework regarding knowledge sharing by Carlile (2004) 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 (see figure 2).

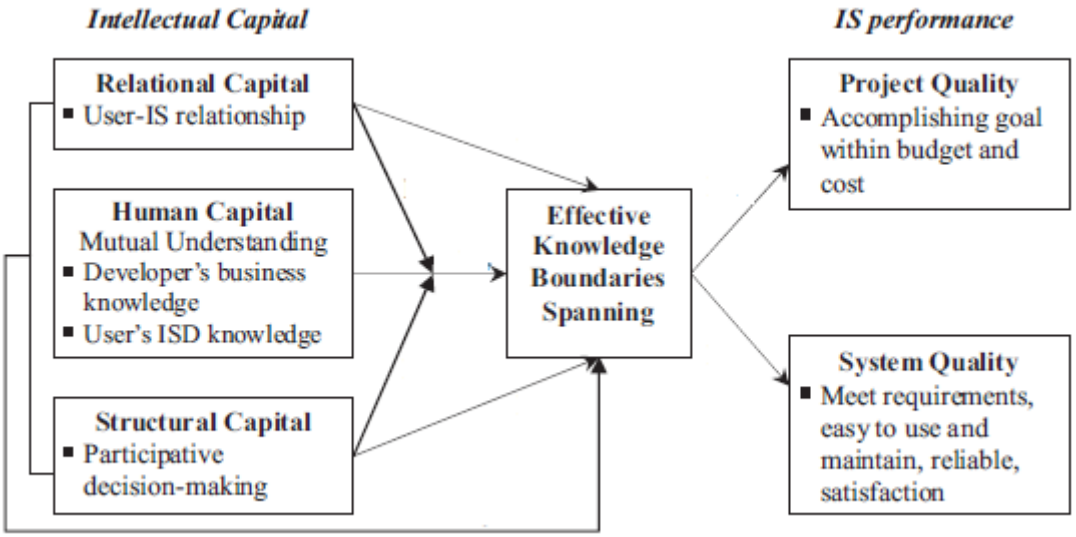


Figure 2, Extended framework as proposed by Hsu, Chu, Lin and Lo (2014).

The three aspects related to intellectual capital are relational capital, human capital and structural capital. Relational capital is concerned with the value of the relationship of the relevant stakeholders of the information service project. Important aspects of this concept are: mutual trust and respect, friendship and reciprocity. Human capital is concerned with the capabilities of the people involved in the project, in the current study it is concerned with the ability of different stakeholders to generate an effective outcome. Their ability to do this is also related to their knowledge and their ability to understand each other and to generate a shared understanding. The third aspect, structural capital, is related to the organizations routines and structures and their effect on interactions between different stakeholders. An example of such a structure/routine is participative decision-making in order to enhance interactions among different stakeholders. The model proposes that intellectual capital has a positive effect on effective knowledge boundaries spanning, which, in its turn, is supposed to have a positive effect on project quality and system quality. Whereby project quality is related to the efficiency of the project (budget- and time related) and system quality is related to the quality of the information system (the degree to which it meets requirements and the level of user satisfaction for example). They tested their model on a Taiwanese sample of information system projects using a survey and they concluded that effective boundary spanning does have a significant effect on information system quality. Secondly, the results also revealed a strong, significant effect of both human- and relational capital on effective boundary spanning (the effects of structural capital were only marginal). Apparently, building a relationship between the different stakeholders and attaining a common understanding is important for

achieving effective boundary spanning and therefor for achieving effective communication and project quality and efficiency. These results could be explained by considering shared mental models. Since the concept of human capital is related to a mutual understanding between two people communicating it seems very similar to shared mental models between two people. This analogy could mean that an increase in shared mental models could cause more effective boundary spanning. This hypothesis is in line with the study by Liao et al. (2014) who found that transactive memory systems increased the quality of communication between different team members.

2.5 Organizational design and software development

The coordination between the RE process and the development process and its effect on the success of a project can also be analyzed from an organizational design perspective. In order to reduce the complexity associated with software development, modules can be created (Parnas, 1972). Modular organizations are organizations that have replaced their traditional form of hierarchy with (smaller) autonomous organizational units (modules) (Baldwin & Clark, 2000). Benefits associated with a modular design are an increase in the manageability of complexity, ability of different parts of a large design to work concurrently, accommodation of uncertainty, the creation of design option and it can be used as a growth strategy to maintain innovativeness (Baldwin & Clark, 2000; Winkel, J.W., Moody, D.L., & Amrit, C., 2008; Simon, 1996). Further, a module can be defined as "a unit whose structural elements are powerfully connected among themselves and relatively weakly connected to elements in other units." An important characteristic of a modular organization is, thus, that the interdependencies between the different modules are weak and architects responsible for designing the modular organization should investigate parameter interdependencies and address the following design rules:

1. Architecture, what modules will be defined and what will their roles be?
2. Interfaces, how will the different modules interact?
3. Integration protocols and testing standards, how will the system be assembled and how will it be tested?

Ernst (2006) 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. Further, he proposed that codification is not sufficient when technologies are changing fast and unpredictably. Like mentioned earlier, an agile environment is characterized by changes, adaptability and flexibility corresponding to changing customer demand. Arguably, then, the application of a modular organizational design might be limited in an agile environment due to its characteristics. The application of a modular design in software development in an agile environment has been noted by Hoda, Noble and Marshall (2011). Besides studying the consequences of lacking customer involvement, they also reported 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. A study by Frank and Hartel (2009) 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. Further, the teams responsible for constructing user stories were working, at least, one sprint ahead of the development teams. This separation was experienced by the development teams as a "mini-waterfall" which had a negative effect on team morale and caused a lack of ownership. In order to address these issues, feature teams were created which consisted of both interaction designers and business analysts (READY) and developers (DONE). These feature teams, thus, both worked on writing user stories and building software meaning that both modules were brought back together to work as one system. This gave business analysts and user interaction designers the opportunity to work in sync with the developers. 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, confirms the hypothesis that the application of a modular design in an agile software development context might not be optimal.

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 (Erich, Chintan & Daneva, 2014). Although adequate (qualitative) academic research regarding DevOps is still lacking, a review study by Erich, Chintan and Daneva (2014) was able to draw some conclusions with respect to this conceptual framework. By integrating development and

operations of information systems, effective and efficient collaboration between both departments can be enhanced. This, in turn, could lead to an increase in operations performance since it fosters continuous development, quicker releases to the end customer and it has positive effects on quality assurance. Further, DevOps was supported by an environment of collaboration and information sharing (among others). The culture that was associated with DevOps is one of open communication, responsibility alignment and trust. This new development thus seems to advocate more integration with respect to (all) the functions related to information systems as opposed to a more modular organizational design. 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.

Like mentioned above, the inappropriateness of modularity can be explained by the fact that it can be argued that these parts are not loosely coupled systems with weak interdependency since customer involvement is highly important in requirements engineering which indicates that, in case of the READY part and DONE part, both modules are highly dependent and, thus, tightly coupled (Hoda, Noble & Marshall, 2011). 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. But what factors could explain this negative effect of modularity on the communication between tightly coupled modules? The fact that modules each have their own goals could cause two different modules to become highly differentiated. This is clearly evident in the study by Frank and Hartel (2009), one of the modules was responsible for writing user stories while the other module was responsible for actually developing the software. In line with the findings by Levesque, Wilson and Wholey (2001) it could be hypothesized that the development of shared mental models is inhibited between two modules due to their differentiated role. This lack of shared mental models between modules, which will be regarded to be very similar to a shared understanding, could be caused by a semantic boundary. In line with Carlile (2004) and Kleinman and Serfaty (1989) the shared mental models caused by a semantic boundary

could cause in decrease in effective communication. These assumptions lead to the following working hypotheses:

H1a: A modular organizational design leads to a semantic boundary between the modules.
H1b: The semantic boundary leads to a lack of shared mental models between the modules.

Another factor that might contribute to the negative effect of a modular design on communication and coordination between modules is the pragmatic boundary as proposed by Carlile (2004). Like mentioned before, a pragmatic boundary occurs when the communicators have different or even conflicting interests. The different goals of the different modules might lead to different interests of the modules which might not necessarily be aligned. The next hypothesize is thus that a modular design leads to (possibly) conflicting interests between modules thereby creating a pragmatic boundary. The emergences of a pragmatic boundary might in turn cause a decrease in effective communication.

H2a: A modular organizational design leads to a pragmatic boundary between the modules.
H2b: The pragmatic boundary leads to a decrease in effective communication between modules.

A final factor that might influence the communication between modules is relational capital. Like mentioned before, relational capital is concerned with the quality of the relationship between communicators. It could be hypothesized that splitting a department into (relatively) autonomous modules leads to a decrease in the relationship between members of the different modules. Therefore, the final working hypotheses of the current study are:

H3a: A modular organizational design leads to a decrease in relational capital between the modules.
H3b: The decrease in relational capital between modules leads to a decrease in effective communication between the modules.

Thus, the current study will address to what degree a modular organizational design can be applied within an agile software development context. It is proposed that effective application of a modular design in such an environment is limited due to interdependencies between the

different modules based on the results by Ernst (2006). This assumption leads to the overall and final working hypothesis:

H4: Effective application of a modular design is limited in an agile software development context.

More specifically, a modular design is hypothesized to lead to a decrease in effective communication between modules due to the presence of a semantic boundary, a pragmatic boundary and a decrease in relational capital caused by the modular design. The research question that the current study will answer is thus:

"To what degree can a modular organizational design be applied in an agile software development context?"

The 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 environment

In order to be able to understand the teams and the communication between them better, it is important to understand in what kind of environment they operate. The first part of this section will explain the organizational structure of the Rabobank groep" including "Rabobank Nederland", the "ICT groep" and its departments in more detail. Further, it will elaborate on the way agile software development is implemented in the Rabobank, more specifically "CRM Distributie".

3.1 Organizational structure of the "Rabobank groep"

The "Rabobank groep" is a bank providing financial services to its customers. The Rabobank was founded on cooperative principles, meaning (among others) that it is owned by their customers. The Rabobank groep consists of their customers, local member Rabobanks, "Rabobank Nederland", Rabobank International and subsidiaries, see figure 3. The cooperative nature of the bank ensures local independence, autonomy and involvement, whereby "Rabobank Nederland" was founded to facilitate the local member banks by developing policies and products. Further, it supervises the local bank members in behalf of

"De Nederlandsche Bank". The ICT group of Rabobank consists of four departments: "ICT beleid en architectuur (IBA)", "application development and maintenance (ADM), "IT operations (ITO)" and "klantencontact (KC)", see figure 4. The second department, ADM, develops and maintain business applications related to the banking services of the Rabobank and the teams that are being described in the current study are located within this department. ADM works with a customer focus and is divided on a portfolio basis. One of these portfolio's is "CRM Distributie". This portfolio is concerned with better cross-channel services for the customer at lower costs. The final distinction to be made is that of a separate program cluster within the portfolio called "Online". All teams that have been studies are located within "Online". This cluster works on improving the online service for customers and it is concerned with monitoring of the online channel. Further it aims at improving the online banking environment of both mobile devices and stable work areas.

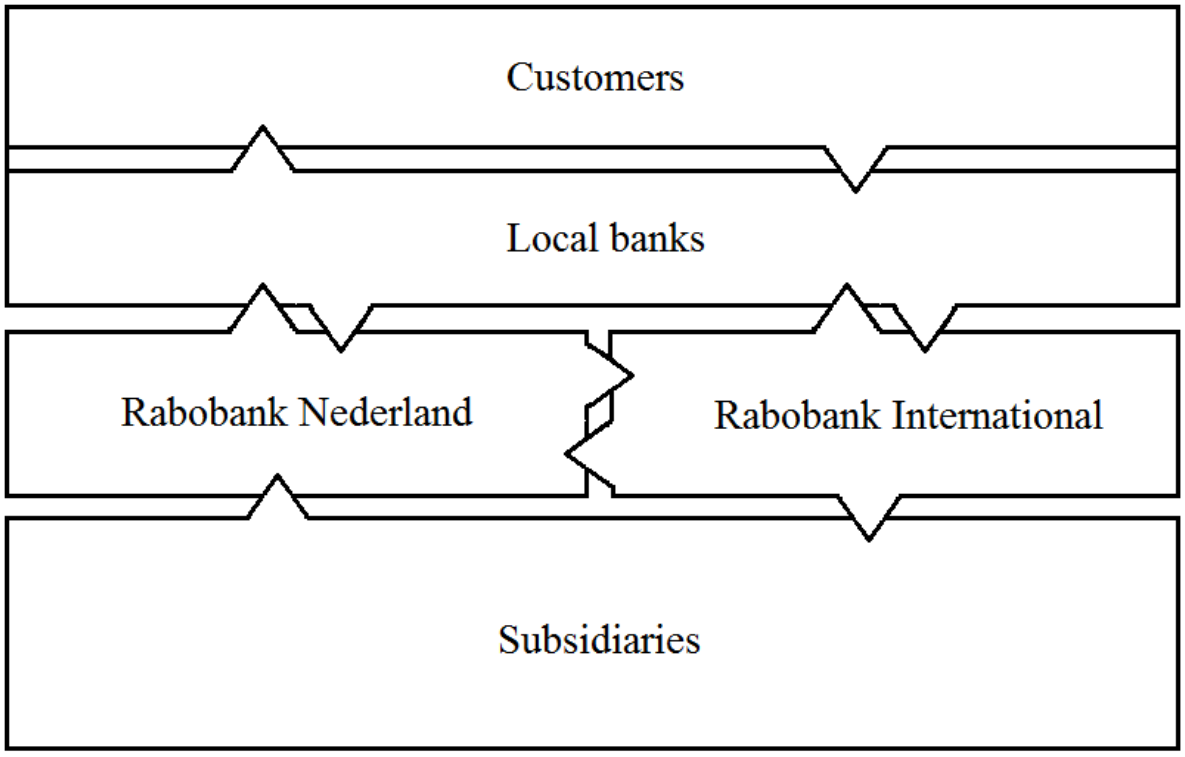


Figure 3, Organizational chart of the Rabobank group.

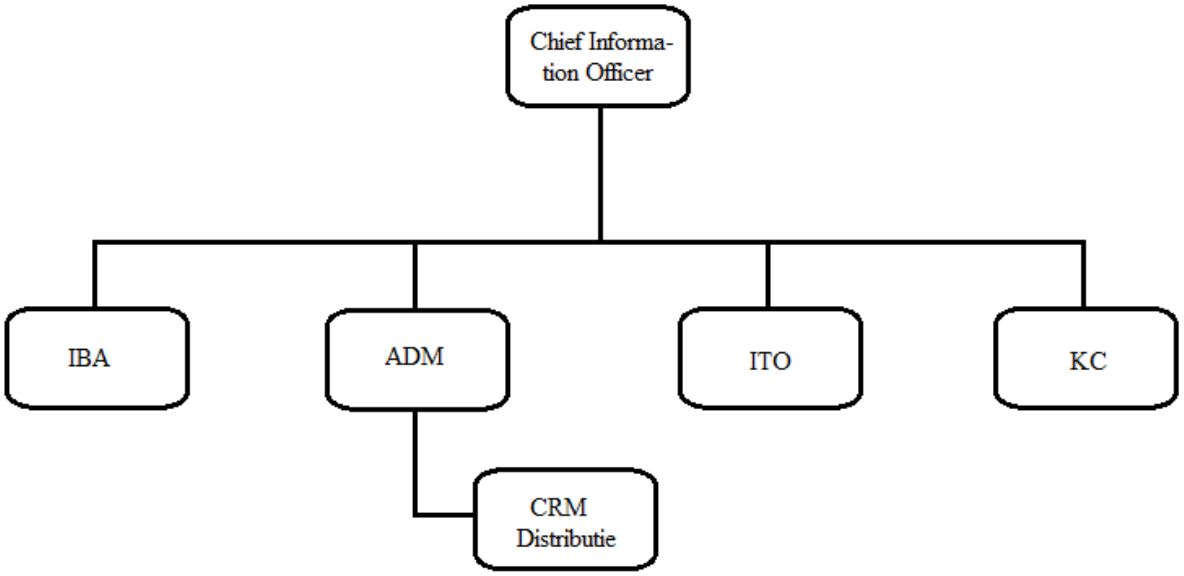


Figure 4, Organization of the "ICT groep" whereby only the "CRM Distributie" portfolio is displayed.

3.2 Agile and "CRM Distributie"

The portfolio, "CRM Distributie", has adopted agile software development practices based on the scaled agile framework and these are referred to by "the agile model". Part of the agile model is a modular organizational design. Two modules were created from the scrum teams, respectively a READY part which is responsible for getting user stories "READY" and a done part which is responsible for getting the user stories "DONE". In the next section, an overview with respect to the implementation of the model is given followed by a section describing both the READY and DONE teams and a section dedicated to the processes.

3.2.1 The implementation of the agile model

The agile model was implemented in program online for several reasons. 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. Another part of this transformation was that one person, the coordinator, was made responsible for all the READY teams and the continuity thereof. It was thought that, by decoupling the DONE teams from a certain project and thus one 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 by coordinated by one person who would

become responsible for the continuity of these teams and, thus, the HR. In line with how the agile model is also called, the factory model, the DONE teams can be seen as a factory that is able to build any user story based on its priority. It is 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 consequences of this arrangement is improved scalability, the program has become more able to respond to changes. When a higher demand for developers occurs, new teams can be created and when demand drops, teams can be send home.

The implementation process started with designing the process. Management, together with the people that were going to be affected by the new agile model and were interested in shaping it, created this new agile model. The design process occurred using working groups, consisting of all people that were interested in designing the process from program online, each responsible for their own theme. In total there were six themes and, thus, 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 process was finished and the new agile process was made definite and implemented. The actual implementation consisted of a presentation to inform the people of the program about the new model.

3.2.2 Description of the teams

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

The so-called "READY teams" are responsible for transforming business wishes into user stories that can be built by DONE teams. These teams consist, generally speaking, of a product manager, project manager, application engineer, test manager, interaction designer

and a business analyst. These READY teams are organized among six theme's: 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 clients world and cross-channel contact. The exact composition of the READY teams is not rigid and varies per theme. Among the READY teams are both internal and external team members who are nationally distributed and do not contain Indian team members.

3.2.3 The agile process within program online

The DONE teams engage in a scrum process, as illustrated in figure 5.

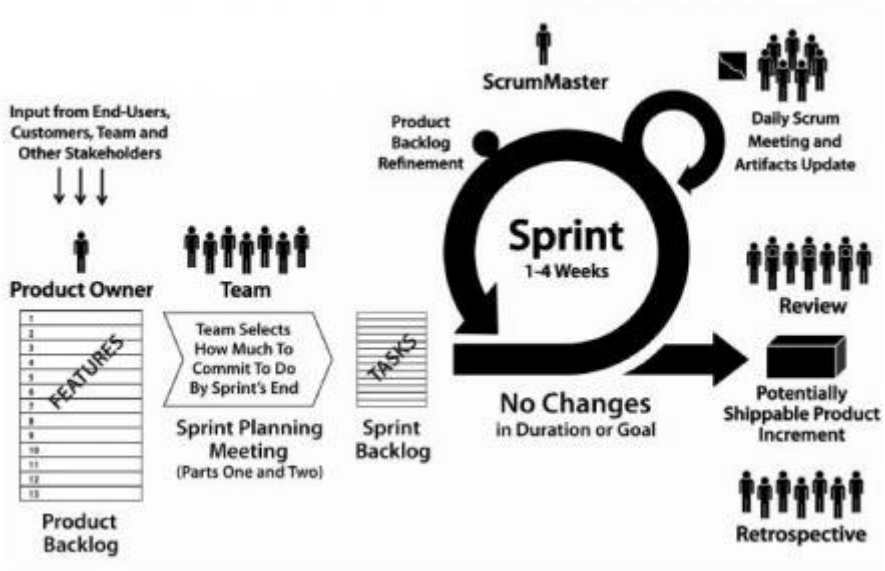


Figure 5, The scrum process as defined by Schwaber (2004).

The first part of the scrum process is concerned with the product backlog which contains all requirements regarding the product that is being developed. The product backlog is prepared by the ready teams who receive their input from other relevant stakeholders like the business or "Exploitative & Beheer Virtuele Kanalen" (EVK) who deal with incidents and issues related to the different channels. The input obtained from relevant stakeholders is turned into user stories which are being "groomed" and pokered together with the done team. During these grooming sessions irrelevant user stories are removed from the backlog, priorities can be changed and estimates can be assigned/altered. The DONE teams are responsible for the sprint backlog, which is prioritized by the product owner. The sprint backlog is prepared during the sprint planning meeting during which the DONE teams commit to a certain amount

of work. A sprint lasts three weeks and during these three weeks the DONE teams engage in daily stand-ups aimed at quickly discussing what each individual member has been working on, will be working on and possible obstacles he or she may encounter. At the last day of the sprint, the software that has been built will be presented during a demo. After the sprint a retrospective meeting is organized with the purpose of evaluating the past sprint.

The READY teams can engage in the DONE process during all relevant rituals: grooming session, daily stand-ups, demo and the retrospective. The commitment of the READY team to the scrum process of the DONE teams varies between the teams and even between the members of one READY team. In general, the READY process starts with a t-shirt planning during which the READY team estimates the size of a project using features. These features are, later, broken down into user stories that can be built by a DONE team.

4. Methodology

This section will describe the methodological aspects of the study. The first part will discuss the research question, followed by an explanation of the research strategy and data sources. Finally, the data analysis approach will be explained.

4.1 Research question

The current study tries to answer the following research question:

To what degree can a modular organizational design be applied in an agile software development context?"

Due to the fact that we are not able to test hypotheses with qualitative research, the hypotheses described below are working hypotheses. Their goal is to guide and structure the investigation. The working hypotheses of the current study are:

H1a: A modular organizational design leads to a semantic boundary between the modules.
H1b: The semantic boundary leads to a lack of shared mental models between the modules.
H1c: The lack of shared mental models caused by a semantic boundary leads to a decrease in effective communication between modules.

H2a: A modular organizational design leads to a pragmatic boundary between the modules.

H2b: The pragmatic boundary leads to a decrease in effective communication between modules.

H3a: A modular organizational design leads to a decrease in relational capital between the modules.

H3b: The decrease in relational capital between modules leads to a decrease in effective communication between the modules.

H4: Effective application of a modular design is limited in an agile software development context.

4.2 Research strategy

The current study uses a holistic, single-case study focusing on the entire program "Online". This selection was made based on the selection criteria proposed by Yin (2003): the research question is a "how" question, it does not require control with respect to the behavioral events and it is concerned with contemporary events. The current study tried to guarantee internal validity by including people with different roles and functions who probably have different views and perspectives. By including different people in the study a bias towards the opinion/perspective of a single function (developer for example) was dealt with. By including three different types of data sources, data triangulation was applied: people, literature and documents (Miles & Huberman, 1994). A more detailed description of the research materials are provided below:

1. People. The study used interviews in order to obtain the perceptions of members of both the ready and done teams as well as line management.
2. Literature. Relevant literature was used to construct a theoretical framework with respect to inter-team communication within an agile software development context. Literature was also used to provide some suggestions with respect to the improvement of the communication between both team types.
3. Documents. Relevant documents like process/role descriptions were used as well as documentation with respect to scrum and the ready/done division.

Yin (2003) proposes three principles in order to deal with problems related to construct validity and reliability:

1. Use multiple sources of evidence. How the current study has tried to conform to this principle is described above.
2. Create a case study database. The interviews were recorded and both the recordings and the transcripts were saved in order to enable other researchers to validate the results. The recordings and transcripts are, however, not included in this report.
3. Maintain a chain of evidence, meaning that another researcher should be able to trace the steps from the conclusion back to the research question and the other way around.

4.3 Data gathering and analysis

This section will describe the data sources used in the current study, how they were gathered and how they were analyzed.

4.3.1 Interviews

The current study uses semi-structured interviews, meaning that some topics were prepared beforehand but to such a degree that there is room left to improvise. Myers and Newman (2007) provide researchers with a framework to conduct semi-structured or unstructured interviews in an information systems context by introducing the theatre analogy for social interactions as proposed by Goffman (1959; 1961). This dramaturgical model and its concepts will be used to provide a rationale with respect to the different aspects of the interview by presenting seven principles to increase performance:

1. Situating the researcher. In order to enable other researchers to evaluate the validity of the results, a part of the paper is dedicated to introducing the interviewer. By providing other researchers with relevant information about the interviewer, they are able to judge what kind of influence the researcher could have had on the interview and the interpretation of the data. The interviewer was a 23 year old, female psychology and business administration student with the Dutch nationality.
2. Minimize social dissonance. The social distance between the interviewer and the interviewee was addressed by not dressing too formal and using jargon that the interviewee used as well.
3. Represent various “voices”. The current study uses “triangulation of subjects” in order to avoid one voice from emerging, by selecting members of both types of teams as well as the relevant line management layer. Further, different functions were included like scrum masters, developers and managers to avoid an elite bias.

4. Everyone is an interpreter. The interviewer tried to evaluate the findings within their context: who said it. A developers view with respect to the process (relatively operational) is probably a lot different from the view of the managers (relatively strategic) for example.
5. Use Mirroring in questions and answers. The interviewer tried to adjust its vocabulary to the interviewee by using similar jargon. Further, the interviewer used question to check whether the interviewee was understood well.
6. Flexibility. The script of the interview only contained main questions/topics to be discussed during the interview in order to maintain some room for improvisation while ensuring that vital topics were still discussed.
7. Confidentiality of disclosures. Interviewees were ensured that all data will be processed anonymously and that no names will be included in the final report. The records were stored and named using a number that was assigned to each interviewee individually and only the interviewer had access to the list regarding what number was connected to which interviewee.

4.3.1.1 The sample

The current study engaged in 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. Program "Online", which was the case study used, contained very diverse projects in terms of complexity ("newness" of the type of programming that the project required, the number of dependencies of the project and the type of dependencies (dependencies across the program, the organization or even other organizations)) and functionality. Due to this diversity, the sampling occurred within the different domains across the program to ensure a diverse sample in terms of the projects the respondents worked at. This was done in order to control for project specific results. Further, a heterogeneous sample was created by including at least one member of the most common functions/roles within the program. These functions are test manager, project manager, application engineer, product manager, business analyst, interaction designer, scrum master, tester, developer and product owner. This was done to control for function related results. By using a heterogeneous sample in terms of projects and functions, the current study is able to identify a general pattern by aggregating the results obtained from the different respondents. After each interview, the

researcher asked the interviewee who would also be interesting to interview, hence the snowballing. This way, the sample would contain the most information rich informants and interviews with people that had only worked at the program for several weeks were avoided. 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 table 1.

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

4.3.1.2 Data collection and registration

All interviews were face-to-face and took place in a familiar location for the interviewee: A meeting room in the office. This location was chosen to increase the interviewee's comfort and to ensure a private conversation. Potential interviewees were invited to participate in the study via email. After confirmation, the participant received an official invite containing information with respect to the duration of the interview, the research goals and the topics that would be discussed during the interview. 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. However, one of the interviewees objected to being recorded and in this case notes were made during the interview. Afterwards, transcripts were made of the interviews.

The script used during the interview can be found in Appendix A. In short, the interviewer introduced herself and her research. After this short introduction, the interview was explained in terms of duration and the structure. Further, the interviewer asked whether the interviewee agreed with recording the interview and the anonymous processing of the data was emphasized. After this introduction, the interview and the recording started. The first part of the interview was concerned with the scrum process within the Rabobank and the second part was used to elaborate further on the ready/done division. After all topics included in the script were discussed, the interview ended. At the end of the interview, the interviewee was asked whether he or she wanted to add something and whether he or she knew someone else that might be interesting to interview.

4.3.1.4 Data analysis process

The data analysis process occurred according to Miles and Huberman (1994) who define three sub processes:

1. Reducing the data refers to selecting, simplifying, transforming e.g. of the raw data (the raw data of the current study are the transcripts and recordings of the interviews). The most important phase of reducing the data is coding of the transcripts. The first step was to get familiar with the data which means that recordings were listened to at least once after the interview had occurred and the transcripts were read at least once after the interviews were conducted. The second step in reducing the data is the construction of an initial coding list. Finally, the codes were applied to the transcripts and if necessary, the code list was adjusted until it appeared to fit the data.
2. Display the data. The data was displayed by creating a network from the relevant codes/concepts that emerged from the interviews and their relationships. The network can be found in Appendix C.
3. Drawing and verifying conclusions. This final process consists of three steps:
    - Look for alternative themes and reflect on their ability to describe the data/results.
    - Review outliers. Look for examples that does not fit the pattern and try to find explanation for these deviations.
    - Triangulate. As mentioned before, the current study used different data sources in order to prevent biases.

All codes were assigned using the software "Atlas TI". ATLAS TI is a software package designed for qualitative researchers and can help to structure and sort the data and help to find

relationships and trends in the data. ATLAS TI can be used as a supportive tool in a Grounded Theory based research. The coding process resulted in 58 unique codes which can be found in Appendix B. The 58 unique codes were connected to each other in a network, which can be viewed in Appendix C.

Coding occurred by assigning anything from words to short sentences to meaningful pieces of transcript. Like mentioned before, this resulted in 58 unique codes. 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 6.

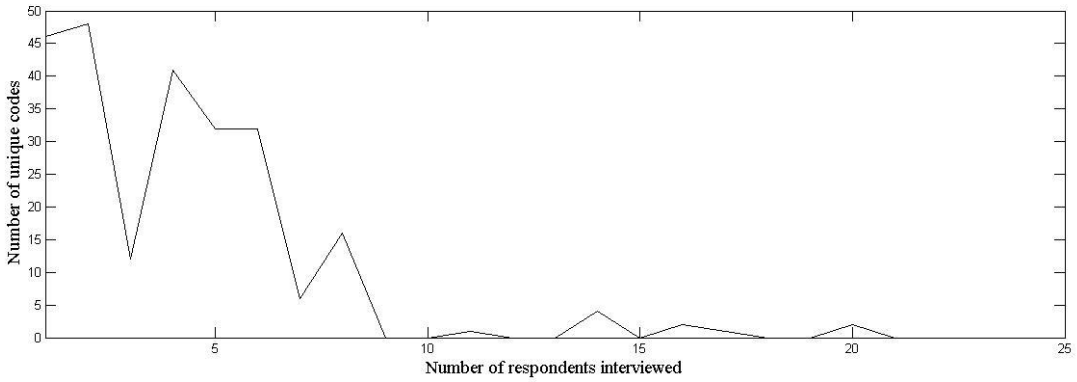


Figure 6 The saturation process.

Figure 6 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. An overview of the relationships between the different codes, see figure. From these clusters of codes, categories with respect to the research question were constructed. In total, four code categories were found: READY process, implementation

process, knowledge management, READY/DONE collaboration and context. An overview with respect to the distribution of the code categories can be found in figure 7.

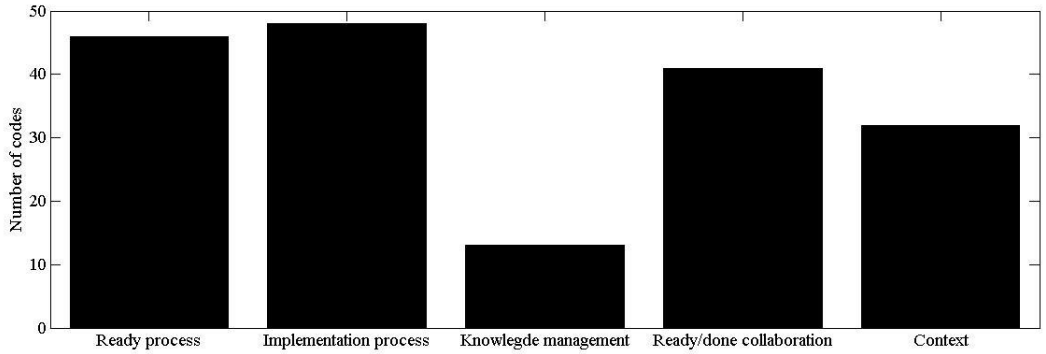


Figure 7 Number of codes per category. Categories from left to right: ready process, implementation process, knowledge management, ready/done collaboration and context.

Figure 7 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. The context of the READY/DONE process was also mentioned relatively often. This can be explained, however, by the fact that interviewees had to provide the interviewer with contextual information in order for the interviewer to really understand the issue the interviewee is describing.

4.3.2 Documentation

This section will describe the documentation that was used in the current study. The majority of the documentation was collected from "SharePoint", which is used by the program to describe the "the agile model" process and the implementation thereof. Systematically, all documents with respect to the implementation process and the agile process were analyzed. With respect to the implementation process, documents were found with respect to the presentation that was given in order to introduce/inform the program about the changes in the process that were about to occur. Further, the people working within the program helped shape the new process in working groups. The documents produced by these working groups were also analyzed.

5. 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 Dutch, the quotes were translated and the original quotes can be found in Appendix D. The results section will explain what the interviews revealed the consequences of the modular design. Finally, the results section was structured using the working hypotheses mentioned in both the literature review and the methodology section.

5.1 Modularity and communication boundaries

5.1.1 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. More concretely, people perceived that two things were lacking: operationalization and evaluation. People perceived the introduction of the model during the implementation to be focused on high-level, strategic implications and lacking operational guidelines. The implementation consisted of a presentation that was given to all members of the program. During the presentation, the philosophy of the new agile model was discussed as well as the benefits it was supposed to bring in terms of flexibility at program level. However, the presentation didn't explain the impact of the model beyond at operational level, so in terms of changes in the different roles (like the business analyst), the requirement engineering process and the development process. Although most of the interviewees agreed that it is not necessary to design detailed procedures with respect to the process since some leeway is desired, some aspects should be agreed upon on program level in terms of the new responsibilities associated with the different roles and best practices for example. One interviewee indicated that they had expected a more active implementation containing change management. The perception that the presentation wasn't sufficient to cover the entire implementation process is illustrated by quote 1. This Test Manager clearly indicates that, although the basic idea of the new agile model was explained during the presentation, what changes would consequently take place on a more practical level wasn't explained causing the people working within the program to be unsure with respect to what was now expected from them within the context of the new agile model. He further talks about the lack of change management during the implementation process as he explains that, besides mentioning the

practical implications of the model during the presentation, these changes should have be implemented more actively by management.

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)

This lack of a clear picture with respect to what consequences the new modular would have on an operational level was also stressed by a Scrum Master that was interviewed (quote 2). He further stresses that the implementation as it had occurred wasn’t sufficient in order for the people working within the program to be able to work with it properly. This inability to work with the new model stems from the fact that it wasn’t clear how the strategic goals are supposed to be achieved on an operational level. In other words, the new requirements engineering and development process associated with the modular design were not sufficiently communicated and implemented.

Q2: “It is all on a very general, high-over level. In terms of strategy (…). But what I always thought was lacking is a good implementation of the strategy and how they pictured it” (PO1 – Scrum Master)

After one of the key figures with respect to the agile model and its implementation was interviewed, it became apparent that the operationalization of the model wasn’t filled in on purpose. The idea was to give people the freedom to find out what worked best for them and to prevent it from becoming a model that was forced onto people. Some interviewees indicated that leeway with respect to the operationalization was, indeed, desirable due to the fact that the different teams are working on very different projects that have different needs. However, they would have expected some kind of evaluation of the model after a certain period of time. This evaluation should entail more concrete things like best practices, what way of working works best and should be used by all teams, but also more strategic things should be discussed like the current status with respect to the implementation of the modular design and were they were heading. The lack of clarity with respect to the modular design is illustrated by an interviewee that indicated that he had heard somewhere that the modular design wasn’t even used anymore. The fact that it isn’t even clear to some people if the modularity is still applied at all is a clear symptom of the lacking managerial communication.

This lack of communication by management is illustrated perfectly by quote 3. This Test Manager indicates that the communication of changes made by management in the strategy of the program aren't explained sufficiently to the people working within the program which often leads to a lack of clarity with respect to what is now expected by management of the people working within the program.

Q3: "I think that the bottom line is that good communication is lacking. (…) Things are decided at managerial level but they aren't communicated downwards in a proper manner." (PO6 – 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 4 by a Business Analyst:

Q4: "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. Besides variation in the interpretation of concepts, the processes employed by the different teams also vary greatly. This was caused primarily by the fact that there is no program wide accepted "READY process" so the methods applied in the READY module vary more greatly compared to the DONE module: the entire DONE module applies the scrum meetings for example but there no agreed upon meetings in the READY module causing one READY team to engage in "kick-off" sessions before the start of every project while other READY teams don't arrange a "kick-off" session at all. These findings confirm the intuition of the working hypothesis that a modular design in an agile environment leads to a semantic boundary between the modules. These results are also in line

with the model as proposed by Hsu, Chu, Lin & Lo (2014) who concluded that a common understanding positively affected effective communication.

Another aspect of the current modular organizational design that has caused a semantic boundary is the fact that no consistent collaboration between both modules occurs. More specifically, this means that there are no consistent collaborations between READY and DONE but different DONE teams work with different READY teams and these collaboration also change over time due to changing priorities and projects. This effect was illustrated very well by a Project Manager (quote 5). In his case, there actually were relatively stable collaborations between one READY team and two DONE teams. He observed that this stable relationship led to a common understanding between the different teams as opposed to teams that did not enjoy stable collaborations. Those teams had to invest in building on a shared meaning before they could understand each other sufficiently and work together smoothly.

Q5: "When you have worked together often you understand each other's language perfectly. You do not have to explain every time what a certain concept means. When I am building a function and I need different teams and I do not know them very well, than I have to invest more time in understanding each other." (PO 17 – Project Manager)

A similar issue was stressed by a Scrum Master (quote 6). He had experienced situations whereby members of the READY team did not have the same knowledge and understanding with respect to certain systems and portlets as compared to the DONE team this READY member had to work with. Instead of trying to reach a common understanding, in his experience, sometimes these READY members just made (faulty) assumptions. As a consequence, user stories are being written that are not appropriate and so the DONE team cannot accept it and have to send it back to the READY team during the grooming sessions. His experience, thus, seems to indicate that this semantic boundary is not spanned very actively all the time which inhibits the occurrence of a common understanding and shared mental models across teams.

Q6: "What I am noticing now is that, whenever we need a BA [Business Analyst] that does not know exactly how the system works, is that he starts guessing or he starts making assumptions about how things work. While we, as a team [DONE] can say immediately that it does not work that way. Than we have to send it [user story] back with the message that he

should take another look at it. You would expect that a BA [Business Analyst] would know how such a portlet works and is able to conduct a proper analysis for the business." (PO 15 – Scrum Master)

An Application Engineer further mentioned (quote 7) 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.

Q7: "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 has 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, has caused a decrease in the effective communication and coordination between teams. The interdependencies between the teams of the READY and DONE module create the necessity to coordinate and communicate between them. However, effective communication and coordination is inhibited by the lack of shared mental models since a shared understanding has to be reached before any coordination can occur.

5.1.2 The pragmatic boundary

Another topic discussed during the interview is the split between the READY and DONE teams. More specifically, the different responsibilities were discussed as well as the consequences of discriminating both parts compared to the situation whereby both parts collaborated together in a team. Despite the fact that a lot of uncertainty exists with respect to the new modular design, most of the participants were able to describe the different responsibilities of the READY module as well as the DONE module and how they relate to

each other very clearly and appear to be in line with the official definitions of both READY and DONE. This, at least theoretical, strict division in responsibilities appears to be maintained in practice as well in line with the definition of Baldwin & Clark (2000) of a module which states that a module should have its own goal(s) and be able to work autonomously. An important consequence of this strict division of responsibilities is the lack of a common accountability with respect to the outcome of the sprint. As quote 8 and 9 indicate, a clear division between the responsibilities of READY and DONE is experienced in practice as well. Although both modules work on the same product, this common goal/responsibility is not experienced in practice. When, for example, a new functions for the mobile application has to be developed the READY module solely feels responsible for writing the user stories and after they are finished, their responsibility for the application ends. The same can be applied to the DONE module: they are merely responsible for actually developing the application and they do not feel responsible for getting the user stories finished.

Q8: "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)

Q9: "When we [DONE] are finished with developing a button, we should validate it with UX [READY]. The button is finished, is this what you had in mind? And that does not always happen" (PO 12 – Developer)

This modular design and the relatively strict division of the responsibilities of both modules is also perceived to be very "waterfall like" by some of the participants in line with the observations of Frank and Hartel (2009). 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 10. In his opinion, the modular design with READY and DONE was a waterfall approach. He further mentions that he feels that the user stories are "thrown over the wall" by READY and that DONE has to figure out what has to be done.

Q10: “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)

In a similar fashion, a business analyst felt that this “waterfall-effect” also occurred the other way around (quote 11). In her opinion, DONE should be involved in writing user stories in order to enhance and maintain the quality of the user stories. However, due to fact that the separation of responsibilities appears to inhibit collaboration, DONE is not

Q11: “DONE should, in my opinion, be involved in writing the requirements. Due to the separation [modular organizational design] they [DONE] do not meddle with the user stories anymore.” (PO 2 – Business analyst)

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. In this case, the conflict in interests arises from the fact that READY has no responsibility with respect to the end product of the sprint resulting in a decrease in willingness to collaborate closely with DONE and track their progress. The same reasoning can be applied to DONE, since they are no longer responsible for writing user stories their interest in helping READY getting the user stories READY has decreased which increases their tendency to “throw them back”. Apparently, the pragmatic boundary primarily decreases the willingness/need to collaborate resulting in a decrease in communication. Another factor that was used to explain the pragmatic boundary was the fact that READY and DONE are not working in parallel but sequential. READY first writes user stories and after they are finished, DONE will build them. During the development process, however, READY is already working on something else which could decrease their interest in the user story that DONE is working on since that is already finished for READY. The latter cause of the conflicting interest was also acknowledge by several interviewees although the former reasoning (“throw-it-over-the-wall” effect) was mentioned more often.

These results confirm the intuition that a modular design in an agile environment causes a pragmatic boundary between the modules and that this pragmatic boundary has a negative effect on the communication between the modules. These results are in line with the intuition

of the working hypothesis that a modular organizational design causes a pragmatic boundary between the modules. Again, this result is also in line with the model by Hsu, Chu, Lin & Lo (2014) who showed that different interests negatively affect effective communication. This result is also in line with the literature on DevOps which stresses the necessity to create a culture whereby the interests of the different functions involved in information systems should be aligned in order to foster collaboration.

In conclusion, 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.

5.3 Intellectual capital: the state of relational capital

Besides the "throw it over the wall" effect, the modular design also has 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 12 by a business analyst illustrates what the effect of the modular design is on the relationship between the ready and done modules.

Q12: "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 13). Apparently, the modular design doesn't merely cause an "us and them" feeling which negatively influences the relationship between members of both modules but this also affects the way members of different modules interact with each other compared to the interaction between members of the same module. One consequence is that members of another module are approached differently. From the results it appeared that this meant that members of different teams were approached less often and if they were approached this occurred less often face-to-face compared to members of the same module.

Q13: “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 14) 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.

Q14: “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 Hsu, Chu, Lin & Lo (2014) who found that a decrease in relational capital negatively affect communicational effectiveness which results in a decrease in project efficiency. This inefficiency was also stressed by one of the interviewees who referred to the former situation (before the modular design) as an “oiled machine” which, according to him, was lacking in the current situation. Another consequence of the deterioration of the relationship between READY and DONE is that not every DONE member is acquainted with the READY members responsible for one of the user stories they are supposed to build. Apparently, an intractability problem has developed between both modules as well.

However, the decrease in communicational effectiveness wasn’t experienced by all of the interviewees. After investigating their situation further, a common factor between these interviewees is one-to-one relationship between one READY and one DONE team as well as

physical collocation. Physical collocation, in this context, refers to READY and DONE teams actually sitting next to each other or even mixed (quote 15). A Business Analyst observed that a one-to-one relationship between one READY team and one or more DONE teams created an environment in which the quality of the relationship between the READY and DONE module remained despite the modular design. The physical collocation further appeared to lower the threshold for people to engage in face-to-face communication.

Q15: "What can be observed at CCF [program theme] since they do have that one on one relation after all, is that they sit mixed in with one another. (…) So, no communication barrier will be present there for example". (PO 8 – Business Analyst)

Another quote (quote 16) also illustrates the effect of a one-to-one relationship. In this case, there was a READY team that worked relatively durable together with two DONE teams. Between these teams, relatively stable relationships between members of different teams have been able to develop. Consequently, the product manager describes how these inter-team relationships have fostered effective communication and collaboration between these teams.

Q16: "That wall that has been planted after the implementation of READY and DONE is broken down a little by our people." (PO 21 – Product manager)

Another aspect that was associated with these teams was that they, besides a one-to-one relationship and physical collocation, had made one of the members of READY responsible for a successful end of the sprint along with the DONE team. 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 result is in line with the study conducted by Hsu, Chu, Lin & Lo (2014). 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.

In conclusion, the current study has provided evidence with respect to the negative effect of a modular organizational design on the relational capital between members of different modules. Further, this decrease in the quality of the relationship between members of different modules also has a negative effect on the effective communication between members of different modules. Due to the fact that human capital does not differ significantly from the concept of shared mental models, this concept will not be discussed.

6. Conclusion and discussion

6.1 Discussion

6.1.1 Limitations

Like mentioned in the methodology section, the current study uses semi-structured interviews. Besides several benefits that are associated with this type of interviews (lack of pre-judgment of the interviewer which enables the interviewee the talk about things that he/she thinks are relevant/important in depth with the potential of increasing validity for example), this method is also associated with several drawbacks (Baarda, De Goede & Teunissen, 2009; Miles & Huberman, 1994). One of these drawbacks is the lack of comparability between the individual interviews due to the fact that respondent might be answering different question (although talking about the same topics). The current study dealt with this by (anonymously) confronting interviewees with quotes or conclusion by other interviewees. This increased the comparability of the data by directly addressing this issue during the interviews: if the researcher thought that the remark by an interviewee was related to certain topics discussed during other interviews, the current interviewee was asked if he or she shared the opinion by another interviewee. The interviewer further asked the current interviewee why he or she could or couldn't relate to the statement provided by the interviewer which helped to gain a deeper and more nuanced understanding of the topic discussed. The data resulting from semi-structured interviews can also be different to analyze due to the often rich and in-depth data that is obtained. A difficulty that arises is concerned with deciding what is important/the core and what is less important and too specific/detailed. This issue was addressed by actively comparing the data and codes attached to the transcripts of the different interviews. By comparing the individual instances of communication issues between READY and DONE provided by the different interviewees for example, recurring themes and concepts could then be identified as well as the relationships between these concepts. This active comparison

helped to discriminate the important data from the relatively less important data causing a general theory to emerge. A final difficulty is concerned with the validity of the results: although semi-structured interviews possible increase the validity in comparison to structured interviews it is difficult to check whether this is actually the case.

The internal validity of the current study was guarded, first of all, by transcribing all the interviews and by storing them as a reference. This was done to address the threat of the interview techniques of the interviewer. Further, construct validity was also considered by saving a chain of evidence (Yin, 2003). Data source triangulation was used in order to be able to avoid role specific biases. The current study uses data source triangulation by including evidence from related literature, documentation within the organization and interviews. These different sources of data can further be used to triangulate conclusions based on the interviewees. Triangulation was also applied to the data sources (interviewees) used for the interviews: members from READY, DONE and line management were all included in the interviews. Triangulation of interview sources was addressed in order to limit group specific biases. Further, a protocol was used in order to create similar circumstances across the different interviews. A final aspect that was addressed was the influence of the interviewer on the interviewee. The current study tried to make the interviewee feel at ease as much as possible by conducting the interviews in a private meeting room in a familiar environment (the organization). When the interviewer felt that the interviewee felt uncomfortable, the first part of the interview protocol (introduction of the interviewee) was elongated. Generally, interviewees became more comfortable when talking about their role in the organization a little longer before the actual interview started.

The external validity was improved by conducting replication logic in line with Yin (2003). By using different types of data sources and including interviewees from different project, the current study tried to conduct multiple case studies within the case study thereby trying to establish that external influences will cancel out and a general theory will emerge.

6.1.2 Recommendations for future research

Due to the fact that the results of the current seem to fit within the emerging literature with respect to DevOps it might be interesting to see whether these conclusions actually hold in a DevOps environment. Therefore, replicating the current study in a DevOps environment might give interesting results.

Future research could conduct a quantitative study to verify the results of the current study and draw conclusions with respect to the validity of the working hypotheses that were developed. More specifically, quantitative research could draw conclusions with respect to whether the hypotheses can be confirmed or should be rejected. Another aspect that would be interesting to investigate is whether a decrease in effective communication due to a semantic and a pragmatic boundary and a lack of relational capital also negatively affects efficiency. Despite the fact that interviewees sometimes indicated that they felt that efficiency had decreased after the modular organizational design was implemented, the current study was not able to draw hard conclusions on this effect due to a lack of available data.

6.2 Conclusions

6.2.1 The effects of modularity on effective communication and coordination

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). 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 Carlile (2004),

Hsu, Chu, Lin and Lo (2014) and Espinosa, Kraut, Slaughter, Lerch & Hersleb (2001). Further, the results of the study appear to confirm the findings by Liao et al. (2014). 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 Al-Rawas and Easterbrook (1996). 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. Apparently, 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 Carlile (2004). 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 Hsu, Chu, Lin and Lo (2014). 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. The aforementioned conclusions result in the model as displayed in figure 8.

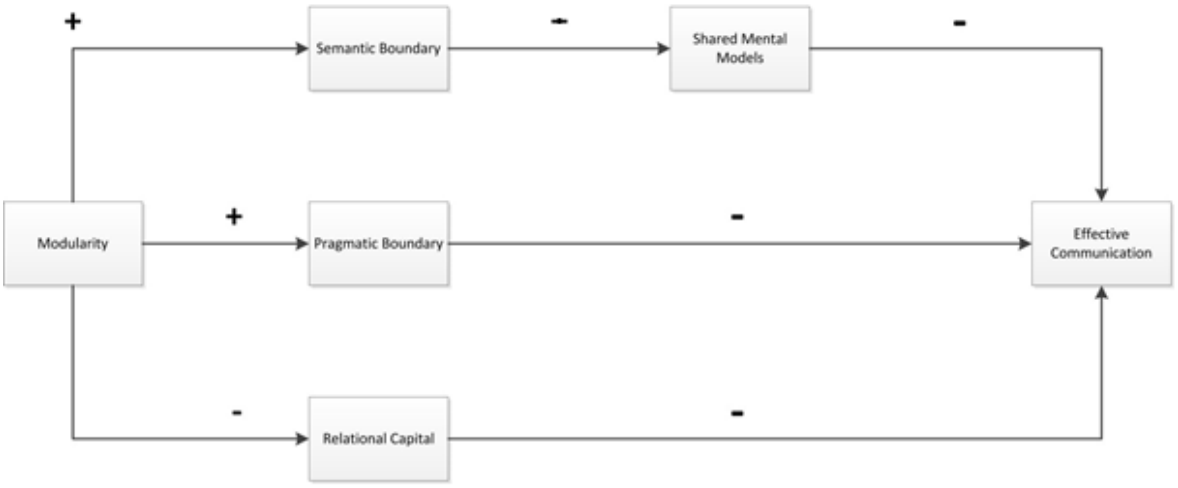


Figure 8 The effect of modularity on effective communication between models.

6.2.2 Managerial implications

The results of the current study revealed that a modular organizational design has a negative effect on the communication and coordination between members of different modules. This effect poses a limit to the application of a modular design in a context of highly interdependent systems. Managers who operate in an agile software development environment have to carefully weigh the benefits of increased manageability of the system through

modularity against the consequences of decreased communication and coordination in a highly interdependent, agile system.

Further, the current study provides evidence that interdependent modules have to be coordinated explicitly due to the fact that an important precondition of implicit coordination, the occurrence of shared mental models among coordinators, is not satisfied satisfactory in a modular organizational design. However, the effectiveness of explicit coordination in an agile software development environment might be limited due to the fact that agile software development relies primarily on the exchange of tacit knowledge which, by definition, has to be coordinated implicitly (Rico et al., 2008).

References

Al-Rawas, A., & Easterbrook, S. (1996). Communication problems in requirements engineering: A field study. *Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering, London.*

Amrit, C. (2008). Improving coordination in software development through social and technical network analysis. Retrieved from: http://doc.utwente.nl/60163/1/thesis_Amrit.pdf

Amrit, C., & Van Hillegersberg, J. (2008). Detecting coordination problems in collaborative software development environments. *Information Systems Management, 25(1),* 57-70.

Amrit, C., Van Hillegersberg, J., & Kumar, K. (2012). Identifying coordination problems in software development: finding mismatches between software and project team structures. *arXiv preprint arXiv:1201.4142.*

Baarda, D.B., De Goede, M.P.M, & Teunissen, J. (2009). *Basisboek Kwalitatief Onderzoek.* Houten: Noordhoff Uitgevers.

Baldwin, C.Y. and Clark, K.B. (2000). *Design rules. Volume 1: The Power of Modularity*. Cambridge: MA.

Beaumont, S. (2010). The definition of ready. Retrieved from: http://blog.xebia.com/2009/06/19/the-definition-of-ready

Blau, P., & Scott, W. R. (1962). *Formal Organizations*. San Fransisco: Foresman.

Brooks, F.P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer 20(4),* 10-19.

Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems, 18,* 332-343.

Carlile, P.R. (2004). Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization Science, 15(5),* 555-568.

Cataldo, M., Hersleb, J.D., & Carley, K.M. (2009). Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity.

Charmaz, K. (2007). *Constructing grounded theory: a practical guide through qualitative research.* Sage: Thousand Oaks.

Cohen, D., Lindvall, M., & Costa, P. (2004). *Advances in Computer: Advances in Software Engineering.* Elsevier: Amsterdam.

Colfer, L., & Baldwin, C.Y. (2010). The Mirroring Hypothesis: Theory, Evidence and Exceptions. *Working paper*.

Conway, M. (1968). How do committees invent? *Datamation, 14(4),* 28-31.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A Field Study of the Software Design Process for Large Systems. *Communications of the ACM, 31(11),* 1268-1287.

Elsevier (2014). “Overheid verspilt miljarden euro's aan mislukte ICT-projecten”. Retrieved

from: http://www.elsevier.nl/Economie/nieuws/2014/4/Overheid-verspilt-miljarden-euros-aan-mislukte-ICT-projecten-1509910W/

Erich, F., Amrit, C., & Daneva, M. (2014). Report: DevOps literature review. Retrieved from: http://www.utwente.nl/bms/iebis/staff/amrit/devopsreport.pdf

Ernst, D. (2006). Limits to modularity: Reflections on recent developments in chip design. *Industry and Innovation, 12(3),* 303-335.

Espinosa, J.A., Kraut, R.E., Slaughter, S.A., Lech, J.F., & Hersleb, J.D. (2001). Shared Mental Models, Familiarity and Coordination: A Mulit-Method Study of Distributed Software Teams. *Proceedings of the 2002 International Conference on Information Systems.*

Frank, A., & Hartel, C. (2009). Feature teams collaboratively building products from READY to DONE. *Proceedings of the ninth agile conference,* 320-325.

Goffman, E. (1959). *The presentation of self in everyday life.* London: Penguin.

Goffman, E. (1961). *Encounters: Two studies in the sociology of interaction.* Indianapolis: The Bobbs-Merrill Company.

Herbsleb, J. D., & Grinter, R. E. (1999). Architectures, Coordination, and Distance: Conway's Law and Beyond. *IEEE Softw., 16(5),* 63-70.

Herbsleb, J.D., & Grinter, R.E. (1999). Splitting the Organization and Integrating the Code: Conway's Law Revisited. *Proceedings of the 21st international conference on Software engineering,* 85-95.

Hoda, S., Nobel, J., & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology, 53,* 521-534.

Hossain, E., Babar, M.A., & Paik, H. (2009). Using Scrum in Global Software Development:

A Systematic Literature Review. *Proceedings of the 2009 IEEE International Conference on Global Software Engineering,* 175-184.

Hsu, J.S., Chu, T., Lin, T., & Lo, C. (2014). Coping knowledge boundaries between information system and business discipline: An intellectual capital perspective. *Information & Management, 51,* 283-295.

Kleinman, D.L., & Serfaty, D. (1989). Team performance assessment in distributed decision making. *Proceedings of the interactive networked simulation for training conference.*

Klimoski, R., & Mohammed, S. (1994). Team Mental Model: Construct or Metaphor? *Journal of Management, 20(2),* 403.

Kraut, R.E., & Streeter, L.A. (1995). Coordination in Software Development. *Communications of the ACM, 38(3),* 69-81.

Levesque, L.L., Wilson, J.M., & Wholey, D.R. (2001). Cognitive divergence and shared mental models in software development project teams. *Journal of Organizational Behavior, 22,* 135-144.

Malone, T.W., & Crowston, K. (1990). What is coordination theory and how can it help design cooperative work *systems? Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '90)*, 7-10.

Malone, T. W., & Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computer Surveys, 26(1),* 87-119.

Miles, M.B., & Huberman, A.M. (1994). *Qualitative data analysis: An expanded sourcebook.* London: Sage.

Myers, M.D., & Newman, M. (2007). The qualitative research in IS: Examining the craft. *Information and Organization, 17,* 2-26.

Nerur, S., & Balijepally, V. (2007). Theoretical predictions on agile development

methodologies. *Communications of the ACM, 50,* 79-83.

Nidumolu, S. (1995). The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable. *Information Systems Research, 6(3),* 191-219.

Ovaska, P., Rossi, M., & Marttiin, P. (2003). Architecture as a coordination tool in multi-site software development. *Software Process: Improvement and Practice, 8(4),* 233-247.

Parnas, D.L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM, 15(12),* 1053-1058.

Perry, D.E., Staudenmayer, N.A., & Volta, L.G. (1994). People, Organizations, and Process \ Improvement. *IEEE Software, 11(4),* 36-45.

Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *The Journal of Software and Systems, 82,* 1479-1490.

Polanyi, M. (1966). *The tacit dimension.* Routledge: London.

Ramesh, B., Cao, L., & Baskerville, B. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal, 20(5),* 449-480.

Rico, R., Sanchez-Manzanares, M., Gil, F., & Gibson, C. (2008). Team implicit coordination processes: a team knowledge-based approach. *Academy of Management Review, 33,* 163–184.

Salas, E., Dickinson, T.L., Converse, S.A., & Tannenbaum, S.I. (1992). Toward an understanding of team performance and training. *In R. W. Swezey & E. Salas (Eds.), Teams: Their training and performance,* 3-29. Norwood, NJ: Ablex

Simon, H.A. (1996). *The Sciences of the Artificial - 3rd Edition.* The MIT Press, October 1996.

Schwaber, K. (2004). *Agile Project Management with Scrum.* Washington: Microsoft Press.

Sosa, M.E., et al. (2004). The misalignment of product architecture and organizational structure in complex product development. *Journal of Management Science, 50(12),* 1674-1689.

Standish Group International (1994). *CHAOS: Project Failure and Success Research Report,* Massachusetts: Standish Group International, Inc.

Verschuren, P., & Doorewaard, H. (1998). *Het ontwerpen van een onderzoek, tweede druk.* Utrecht: LEMMA

Wagstrom, P., & Hersleb, J. (2006). Dependency forecasting in the distributed agile organization. *Communications of the ACM, 49(10),* 55-56.

Winkel, J.W., Moody, D.L., & Amrit, C. (2008). Desperately Avoiding Bureaucracy: Modularity as a Strategy for Organisational Innovation. Proceedings of the 16th conference on information systems.

Yin, R.K. (2003). *Case study research: design and methods*. London: Sage.

Appendix A

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 B

Implementation process → communication

Implementation process → communication → strategic

Implementation process → communication → strategic → diversity between teams

Implementation process → communication → strategic → diversity between teams →

inefficiency

Implementation process → change management

Implementation process → change management → diversity between teams → inefficiency

Implementation process → evaluation

Implementation process → inwerktijd

Implementation process → goals model

Ready/done collaboration → different responsibilities

Ready/done collaboration → different responsibilities → "hidden waterfall"

Ready/done collaboration → different responsibilities → "hidden waterfall" → inefficiency

Ready/done collaboration → different responsibilities → "FLOMMEN"

Ready/done collaboration → different responsibilities → "FLOMMEN" → inefficiency

Ready/done collaboration → different responsibilities → non synchronous

Ready/done collaboration → different responsibilities → ownership DONE

Ready/done collaboration → different responsibilities → ownership DONE → morale

Ready/done collaboration → different responsibilities → relationship

Ready/done collaboration → different responsibilities → relationship → intraceability

Ready/done collaboration → different responsibilities → relationship → inefficiency

Ready/done collaboration → different responsibilities → relationship → factory

Ready/done collaboration → different responsibilities → relationship → factory → morale

Ready/done collaboration → one-to-one relationship

Ready/done collaboration → one-to-one relationship → bonding

Ready/done collaboration → one-to-one relationship → efficiency

Ready/done collaboration → one-to-one relationship → sit mixed in

Ready/done collaboration → one-to-one relationship → sit mixed in → efficiency

Ready done collaboration → shared responsibility → efficiency

Ready process → not agile

Ready process → not agile → not fully dedicated

Ready process → not agile → not-fully dedicated → quality decrease

Ready process → not agile → not fully dedicated → inefficiency

Ready process → not agile → not fully dedicated → lead time increase

Ready process → not agile → not fully dedicated → effect on DONE

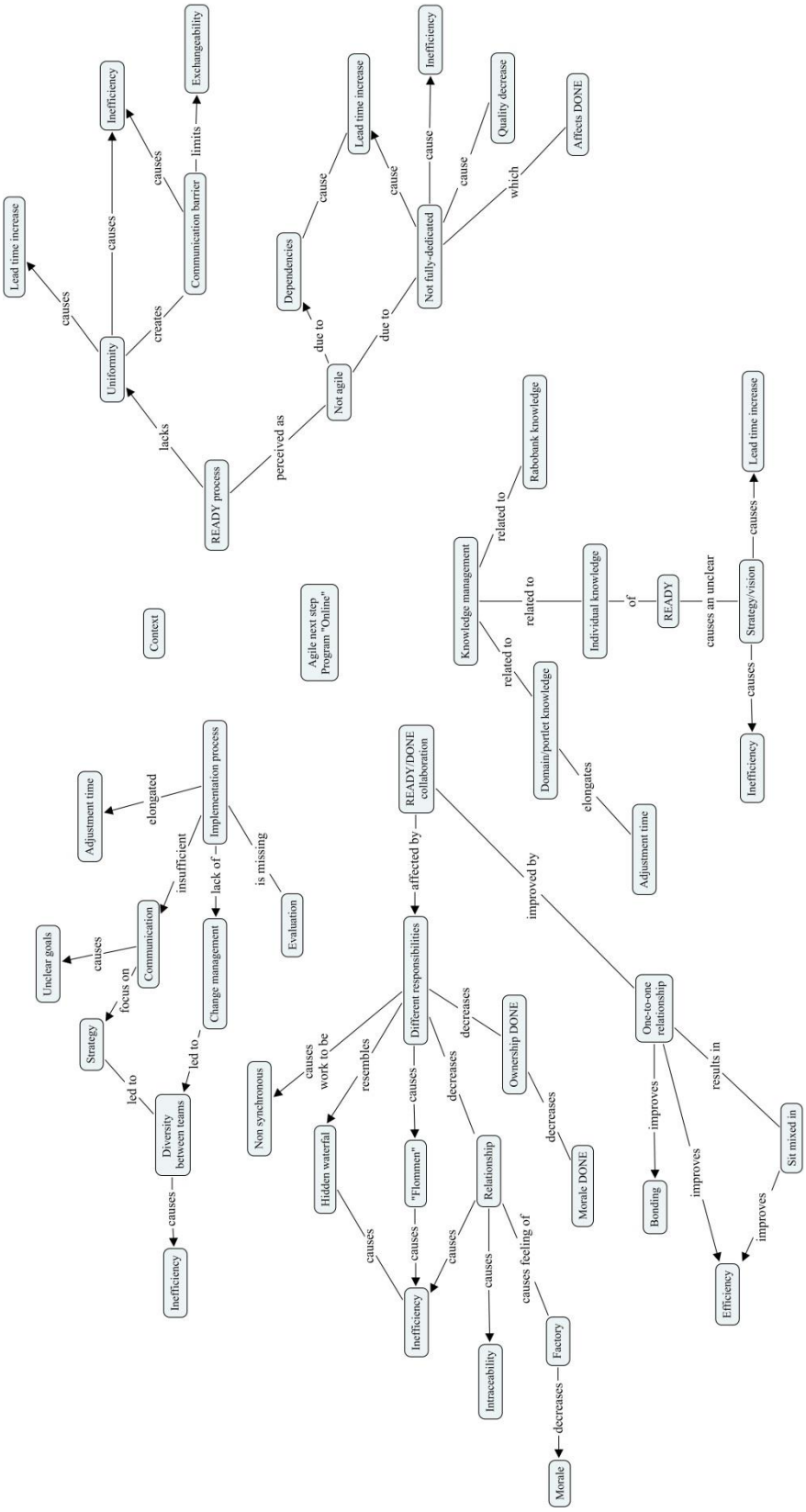
Ready process → not agile → dependencies

Ready process → not agile → dependencies → external parties
Ready process → not agile → dependencies → lead time increase
Ready process → uniformity
Ready process → uniformity → inefficiency
Ready process → uniformity → lead time increase
Ready process → uniformity → communication barrier
Ready process → uniformity → communication barrier → inefficiency
Ready process → uniformity → communication barrier → exchangeability

Knowledge management → "Rabobank knowledge"
Knowledge management → domain/portlet knowledge
Knowledge management → domain/portlet knowledge → "inwerktijd"
Knowledge management → individual knowledge/expertise
Knowledge management → individual knowledge/expertise → READY
Knowledge management → individual knowledge/expertise → READY → strategy/vision
Knowledge management → individual knowledge/expertise → READY → strategy/vision → inefficiency
Knowledge management → individual knowledge/expertise → READY → strategy/vision → project lead time

Context

Appendix C


Lead time increase
causes
Uniformity
causes
Inefficiency
creates
Communication barrier
limits
Exchangeability
causes
lacks
READY process
perceived as
Not agile
due to
Dependencies
cause
Lead time increase
due to
Not fully-dedicated
cause
Inefficiency
cause
Quality decrease
which
Affects DONE
Context
Agile next step
Program "Online"
Knowledge management
related to
Rabobank knowledge
related to
Individual knowledge
of
READY
causes an unclear
Strategy/vision
causes
Lead time increase
related to
Domain/portlet knowledge
elongates
Adjustment time
causes
Inefficiency
Adjustment time
elongated
Implementation process
insufficient
lack of
is missing
Evaluation
Unclear goals
causes
focus on
Communication
Change management
Strategy
led to
led to
Diversity between teams
causes
Inefficiency
Non synchronous
causes work to be
Hidden waterfall
resembles
causes
Inefficiency
Different responsibilities
affected by
READY/DONE collaboration
improved by
causes
"Flommen"
causes
Inefficiency
decreases
Relationship
causes
Intraceability
decreases
Ownership DONE
decreases
Morale DONE
causes feeling of
Factory
decreases
Morale
One-to-one relationship
improves
Bonding
improves
Efficiency
results in
Sit mixed in
improves
Efficiency

Appendix D

Q1: “Ik denk dat je, bottom line is, er wordt niet goed gecommuniceerd. Alles valt of staat met communicatie. Er wordt van bovenaf van alles besloten maar het wordt niet goed gecommuniceerd naar beneden.” (PO7 – Test manager)

Q2: “Een hele mooie presentatie gekregen van dit is waar we naartoe willen en ik heb gewoon heel sterk het idee dat het niet geland is. Tenminste niet voldoende geland is, je moet het wel implementeren.” (PO1 – Scrum Master)

Q3: “Ik denk dat je, bottom line is, er wordt niet goed gecommuniceerd. Alles valt of staat met communicatie. Er wordt van bovenaf van alles besloten maar het wordt niet goed gecommuniceerd naar beneden.” (PO6 – Test Manager)

Q4: “Maar hoe kun jij dan als jij de zwaarte van een feature niet eens kunt inrichten, hoe kun je dan features van verschillende teams met elkaar vergelijken. (…) Dus wordt het heel moeilijk als je een feature plotseling van team een naar team twee moet verplaatsen, en team een en team twee denken anders.” (PO8 – Business Analyst)

Q5: “Wanneer je al heel vaak met elkaar samen hebt gewerkt ken je elkaars taal perfect. Je hoeft niet iedere keer uit te leggen wat het precies is. Als ik functies bouw en ik ben iedere keer bij een ander team en ik ken dat team nog niet zo goed dan moet je meer tijd steken in elkaar blindelings begrijpen.” (PO 17 – Project manager)

Q6: “Wat ik nu vaak merk is dat als we een BA nodig hebben, die weet niet helemaal goed hoe het systeem werkt. Ja die gist een beetje, of die pakt wat punter eruit waarvan hij denkt zo werkt het ongeveer terwijl wij als team gelijk kunnen zeggen zo werkt het niet. Dan sturen we het weer terug en dan gaat hij er nog een keer naar kijken en je zou verwachten dat een BA goed weet hoe zo’n portlet werkt en een goede analyse kan doen voor de business.” (PO 15 – Scrum master)

Q7: “Hoe we het nu hebben moet zo’n DONE team dus in staat zijn om zonder de context die wij [READY] hebben software te kunnen bouwen. Wat uit de praktijk dus blijkt als je dit zo uitvoert [modulaire organisatiestructuur] is dat je heel uitgebreid moet specificeren wil zo’n

DONE team in staat zijn om het in te kunnen schatten en uit te voeren." (PO 20 – Application Engineer)

Q8: "Zij hebben nu niet de verantwoordelijkheid om ook iets op te leveren. Die verantwoordelijkheid ligt nu eigenlijk alleen bij ons. En zij bereiden het eigenlijk alleen maar voor (…) misschien zou dat iets meer gemigreerd moeten zijn. Dat de mensen die de specificaties maken zich ook verantwoordelijk voelen om het op te leveren." (PO12 – Developer)

Q9:"Als we [DONE] met een knop bezig zijn en die is klaar, zouden we dat eigenlijk even met UX [READY] moeten checken. Van hé, ik heb het knopje af, is dit wat je in gedachten had? En dat wordt hier ook niet altijd gedaan." (PO 12 – Developer)

Q10: "READY en DONE is een waterval. Je hebt voorbereidend werk en je gooit het over de muur en het moet gedaan worden." (PO 1 – Scrum master)

Q11: "DONE hoort, naar mijn mening, bij het opstellen van de requirements. Door het opsplitsen [modulaire organisatiestructuur) bemoeien zij zich niet meer met de user stories." (PO 2 – Business analyst)

Q12: "Ik heb geen band meer met die bouwers. Ik zit er nu zo ver vanaf, ik moet gewoon user stories opleveren." (PO8 – Business Analyst)

Q13: "Ik zie UX'ers niet echt als onderdeel van ons team. Je gaat toch anders naar ze kijken en naar ze toestappen." (PO 12 – Developer)

Q14: "In eerste instantie ken je elkaar ook niet en ze zeggen ook wel het maakt niet uit of je een ontwikkelteam op de tweede verdieping hebt zitten en je de trap moet nemen of je zit in India. Als je elkaar niet kent stuur je toch maar liever een mailtje. Je mist dan de hele nuance." (PO 14 – Front end developer)

Q15: "Wat je ziet bij CCF [thema], omdat ze toch weer die een op een relatie eigenlijk, dat je die terug ziet. Die zijn door elkaar gaan zitten (…). Dus daar zal die communicatie kloof bijvoorbeeld niet aanwezig zijn." (PO 8 – Business Analist)

Q16: “Dat muurtje, dat er met de implementatie van READY en DONE is neergezet, dat breken we met de mensen weer een beetje af.” (PO 21 – Product manager)