The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems
promotoren: prof. dr. J.C. van Vliet  
prof. dr. B.J. van den Hooff
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This research has been partially sponsored by the Network Institute and the Dutch “Regeling Kenniswerkers”, project KWR09164, Stephenson: Architecture knowledge sharing practices in software product lines for print systems.

Promotiecommissie:
prof. dr. Darja Smite (Blekinge Institute of Technology)
prof. dr. Julia Kotlarsky (Aston Business School)
prof. dr. ir. Rini van Solingen (Delft University of Technology)
prof. dr. ir. Remko W. Helms (Utrecht University)
dr. Patricia Lago (VU University Amsterdam)


Copyright © 2014, Christina Manteli
All rights reserved unless otherwise stated

Cover design by Larisa Wiegant
Published by Uitgeverij BOXPress, ’s-Hertogenbosch
Typeset in E\TeX\ by the author
Acknowledgements

Several people contributed to the successful completion of my four-years PhD adventure. First, I would like to express my special appreciation to my advisors Hans van Vliet and Bart van den Hooff. Hans, thank you for your unlimited guidance and for allowing me to grow as a researcher. Bart, thank you for all your encouragements and positivism that kept me motivated and self-confident.

To my colleagues Patricia Lago, Maryam Razavian, Klaas Andries de Graaf, Liang Peng, Giuseppe Procaccianti, Damian Andrew Tamburri, as well as the former members of the group Qing Gu and Hector Fernandez: thank you for all the work we’ve done together, but also for all the pleasant moments I shared with you, in and out the office. I would especially like to thank Antony Tang, Rahul Premraj and Victor Clerc for guiding me through the first steps of my PhD. I would also like to thank the former MSc students, Ramon Noordeloos and Sara Bolognezi for contributing to my research. Elly Lammers, Caroline Waij and Mojca Lovrencak have my warmest gratitude, for all their help and administrative support.

The research reported in this thesis is the result of various fruitful collaborations. It has been a pleasure to work together with people from Océ technologies, particularly Robert van Vonderen, John Kesseler and Richard Nap who I thank for their time and effort. I would also like to express my gratitude to Jan van der Meer and Louis Lagendijk for giving me the opportunity to work with Ericsson. Many thanks to the members of the OpeNER project who provided me with a great deal of data and a best paper award!

Last but not least, these pages are dedicated to my son Willem: you’re too young and yet too wise to teach your PhD mama time management, a lesson I will never forget! And I can’t thank more, my caring husband Wilco not only for his unconditional support all these years, but also for his professional guidance and advice. My PhD journey has now come to an end, but it wouldn’t have been possible without the continuous support of my Greek and Dutch family members and friends, especially my mother Anna, my father Alexandros, and my brother Charalampos.

Amstelveen, August 2014
## CONTENTS

5.2 The case study ................................................................. 70  
5.3 Research methodology ................................................... 71  
5.4 Organizational changes ................................................... 72  
5.5 Analyzing the collaboration networks ................................. 73  
5.6 Related work and discussion ........................................... 79  
5.7 Conclusion ................................................................. 80  

6 Introducing Transactive Memory Systems ............................... 81  
6.1 Introduction ............................................................. 81  
6.2 TM structure and social networks ..................................... 82  
6.3 TM processes as a latent variable model .............................. 83  
6.4 Conclusion ............................................................. 85  

7 The Effect of Governance on TMS ........................................ 87  
7.1 Introduction ............................................................. 87  
7.2 Project overview ......................................................... 88  
7.3 Data collection and analysis ........................................... 93  
7.4 Clustering ............................................................... 95  
7.5 Centrality and Boundary spanners .................................... 99  
7.6 Network core ............................................................ 102  
7.7 Lessons learned ......................................................... 104  
7.8 Conclusion ............................................................. 106  

8 The Role of Brokers as Facilitators of TMS ............................. 109  
8.1 Introduction ............................................................. 109  
8.2 Clusters: a collaboration structure ................................... 111  
8.3 The role of brokers ...................................................... 112  
8.4 Research methodology ................................................ 113  
8.5 Analysis & results ....................................................... 115  
8.6 Threats to validity ....................................................... 121  
8.7 Conclusion ............................................................. 122  

9 Transactive Memory before and after Software Transfers ............ 125  
9.1 Introduction ............................................................. 125  
9.2 Software transfers and governance ................................... 126  
9.3 Knowledge transfers and transactive memory ....................... 127  
9.4 The case study .......................................................... 128  
9.5 Changes in governance model ......................................... 129  
9.6 Changes in transactive memory ....................................... 130  
9.7 Discussion .............................................................. 134  
9.8 Conclusion ............................................................. 136
10 Conclusion 139

10.1 Answering the research questions .......................... 139
10.2 Current contributions and Future Directions .................. 144
10.3 Limitations .......................................................... 148
10.4 Epilogue .............................................................. 149

11 Samenvatting 151
Introduction

What kind of decisions do companies take to manage their global software development (GSD) activities? How do people collaborate with their remote colleagues, and how does their collaboration structure affect their ability to know who knows what and who works on which part of the project? These are the questions we focus on and which we further examine in this four-year research program. This dissertation will guide you through the concrete research questions, research methods, observations and conclusions.

1.1 Global software development and governance

For over a decade, global software development (GSD) projects, in which development activities are distributed between geographically distant teams, are the focus of both researchers and practitioners. Apart from the geographic proximity, temporal as well as socio-cultural differences are also identified as barriers to effective collaboration of the teams [2]. Those distances result in several challenges such as miscommunication, delays in decision making and coordination overhead [170].

Global software development has been studied in different contexts and settings. A prominent trend is the applicability of agile methods in GSD settings and whether such an approach will help us overcome existing GSD challenges, or create new ones. In fact, different opinions exist on whether or not agile software development methodologies can flourish in a GSD environment. Agile methods can help improve communication and collaboration in offshore development, which often results in better business / IT alignment and responsiveness to business changes [130]. However, when using agile methods in GSD, the main principles
CHAPTER 1. INTRODUCTION

of agile become more challenging: direct face-to-face communication occurs less often and the team is not co-located anymore [140].

GSD practices have also been studied in the field of software product lines (SPL) and software reuse [167]. Software ecosystems (SECOs), for instance, have emerged as an approach to improve software reuse in global software development [16]. Within software product lines, organizational structures become more complex and geographically dispersed teams have significantly more difficulties in implementing the necessary coordination [18].

Regardless the approach one will take to study global software development, be it in the use of agile methodologies, or in the context of software product lines, the key challenge remains; how to manage and control the global software development activities? As a result, global software governance emerged as a subfield of information technology (IT) governance. Research on software development governance is rather recent, and as Dubinsky et. al [57] note, it is also the result of an increased focus on the human aspects of software development, such as team work and social collaboration. When the software development activities are globally distributed, the need for a clearly defined governance increases [71].

Governance is defined as those arrangements and practices that an organization puts in place to ensure that the activities are appropriately managed [5]. The goal is to ensure that business processes of the software company meet the strategic requirements of the organization [33]. In order to achieve this, software development processes should be aligned with the company’s business goals on a strategic, tactical and operational level. To date, few proposals have been made defining the attributes of a software governance model for distributed development projects, and the coordination mechanisms that such a model should embrace. Ramasubbu & Balan [157], for instance, present a research model on how to create a governance framework for distributed software development, focusing on three stages of a project lifecycle; planning, execution and reflection. In another work, a governance framework is proposed for managing outsourcing engagements, based on organizational structures, joint processes and relationship management functions [71].

Bannerman [5] suggests that software governance may have a functional perspective in terms of what governance does, as well as a structural perspective dealing with what governance looks like. In this thesis, we make our contribution and define a multi-site software development governance model from a structural perspective. Synthesizing existing knowledge on GSD governance and our observations from industrial case studies, we propose a multi-site software governance model based on three aspects; the business strategy that binds the relationship of the remote offices, the structure and composition of the distributed teams and
1.2. COLLABORATION STRUCTURES

the way tasks are allocated across sites.

1.2 Collaboration structures

In the field of computer-supported cooperative work (CSCW), collaboration is defined as the combination of three aspects; communication, coordination and cooperation [98]. During communication people negotiate and make decisions, while coordination is the management of those people and their activities. Cooperation is the joint operation in order to execute tasks [70]. Coordination in global software development is a challenging task. Since the famous Conway’s Law [35], according to which the software architecture of the system should reflect the organizational structure, numerous studies turned their attention towards the collaboration patterns of software development teams (e.g. in [79, 80]).

People still rely heavily on formal structures when taking decisions, but in fact those structures do not represent the underlying collaboration networks. Previous studies suggest that social network analysis (SNA) techniques can be used to reveal collaboration structures that go beyond the organizational charts and the project planning (e.g. [125, 53, 45]). Taking a social network perspective within an organization “renders the invisible, visible” [40].

Social network analysis (SNA) refers to the study of interconnected nodes and their relationships. Wasserman [180] highlights two important characteristics of social networks; (1) actors are viewed as interdependent, rather than autonomous units and (2) the relationships between actors (ties) are channels for transferring resources, such as information. Hence, taking a social network perspective allows us to examine not only the individual attributes of the actors (such as working experience, role, etc.) but also to study the structure of the network created through the relationships between the actors.

In the following chapters, we examine social networks in GSD by analyzing both their structural characteristics, as well as the individual attributes of the network members. We look at the different structures, such as core-peripheries and clusters, and we study their effect on teams’ collaboration. We also distinguish the influential members, especially people who act as boundary spanners bridging the different clusters of the network.
1.3 Transactive memory systems

Different working practices among GSD teams as well as barriers of multi-site collaboration influence team awareness or in other words, to know who knows what, who works in which parts of the project and who is who in the remote teams [44]. In that regard, Wegner [182] defines transactive memory system (TMS) as a shared cognitive system for encoding, storing and retrieving expertise knowledge between members of a group. Particularly, TMS is the collective memory consisting of the combination of individual memory and the communication links between individuals [84]. Consider, for instance, a group of people working for a project. TMS theory recognizes that every member of the team has his/her domain expertise, i.e. he/she has knowledge about a specific part of the project. Based on TMS theory, people working in a team do not need to know everything about the project but they need to know who has the right information when they need it. In other words, team members communicate with each other and through those communication links (transactions) they develop a memory system (a transactive memory system) about who knows what in the team. This memory system helps them share, allocate and locate the right information, when needed.

Empirical research has shown that transactive memory systems may have a positive effect on team performance, facilitate learning, enable creativity and improve members’ satisfaction [149]. Team familiarity, group expertise and communication are among the most influential factors for the development of TMS between interdependent teams and individuals [159]. For instance, Faraj and Sproull [65] found that when team members are able to locate and recognize expertise knowledge, there is a positive association with their ability to meet project goals within time and budget.

TMS theory originated by examining close relationships among married couples [183]. Recent studies extended TMS theory to complex organizational settings (for example in [3 147 92]), as well as in dispersed and virtual teams (for example in [190 116 144 143]). While applying TMS theory on different contexts, several proposals have been made on how to measure transactive memory systems [3 115 190]. Lewis & Herndon [117] suggest that transactive memory can be either assessed directly or indirectly as a latent variable. Direct measures allow us to draw conclusions about transactive memory as a whole, while indirect measures are useful in settings where the transactions (processes) cannot be clearly assessed. The choice on how to measure TMS lies mainly on the case study at hand [117].

In this thesis, we use the theory of transactive memory systems to explore the extent to which distributed members are aware of each others’ expertise, can eas-
1.4. RESEARCH QUESTIONS

ily access each other and how frequent they communicate. Through a thorough amalgamation of existing literature, we propose a two-fold approach in measuring transactive memory systems in GSD; a structural approach, in which SNA techniques are used to capture the structures of the transactive memory systems. And a process approach, where we build a latent variable model to measure the transactions (processes) created within those systems.

1.4 Research questions

Over the years, globalization of software development turned into a common practice. Factors such as the coordination of activities across locations, tasks’ synchronization through different time zones, and the communication between distributed teams became familiar among scholars and practitioners. As research in the specific field continues and strategies in global software development evolve, the interest turns towards the challenges and key issues of managing GSD activities. To date, few proposals have been made defining the attributes of a software governance model for distributed development projects and the coordination mechanisms of such a model.

In order to properly decide how to govern, i.e. how to manage and control GSD activities, we first need to understand the underlying collaboration structures. As Boehm [9] points out “social issues in software engineering will be a challenge in the next years, since the field needs to treat issues beyond the technical side, which requires observing it in another perspective.” By taking a social network perspective, we can reveal collaboration structures that go beyond technical aspects, systems’ architectures or even project planning.

A major challenge in GSD collaboration is the awareness of where expertise resides within the multi-site network, how to access it and how to effectively communicate with the right people. When such processes (transactions) exist in a network, transactive memory systems emerge and they can be explored for their effectiveness in facilitating collaboration. Despite its prominence in social and business sciences, little we know on how to identify, analyze, measure and evaluate transactive memory systems in the field of global software development.

Bringing everything together, we recognize the opportunity to build upon the body of knowledge on global software governance and transactive memory systems by taking a social network perspective. The main research question that drives the rest of the study is defined as:

**Research Question:**  *How does global software governance affect transactive memory systems?*
CHAPTER 1. INTRODUCTION

To begin our analysis, we first need to understand the nature of global software governance. When companies engage into global activities, they have to decide upon several aspects on how to perform such distributed collaborations. For instance, they can choose to work with external service providers, third-party companies in offshore locations or they can choose to create a captive center in a remote location. Either choice involves different legal as well as knowledge management implications for the collaboration of the distributed teams. We therefore need to define those aspects that characterize a global software governance model, including the type of decisions companies need to consider for managing and controlling their GSD activities. In order to achieve this, we ask the following questions:

**RQ.1:** How is governance defined in global software development?

**RQ.1.1** What aspects embody a global software governance model?

**RQ.1.2** What are the implications of governance decisions in global software development?

Moreover, governance decisions aim at the orchestration of GSD teams and their activities. Hence, we need to understand how software development teams are structured across global settings and which aspects affect their collaboration. Social network analysis techniques gain more and more popularity in analyzing collaboration structures, but to date little we know about their applicability in global software development. Such observations pave the way for the following sub-questions:

**RQ.2:** How can we identify and analyze collaboration structures in global software development?

**RQ.2.1** How can we use social network analysis to explore GSD collaboration?

**RQ.2.2** What kind of GSD collaboration structures exist?

Finally, we need to understand transactive memory systems in the context of global software teams. Hence, we look at multi-site teams and explore the processes and transactions that take place, and which enable members to learn about their remote colleagues, become familiar with each other and facilitate finding and retrieving expertise knowledge. This effort leads us to the following sub-question:

**RQ.3:** How are transactive memory systems defined in GSD?

**RQ.3.1** How can we identify and measure transactive memory systems?

**RQ.3.2** What aspects influence the development of transactive memory systems?
1.4. RESEARCH QUESTIONS

1.4.1 Research in a nutshell

This research aims to extend current knowledge on global software development collaboration and particularly on team awareness, by taking a social network perspective. Figure 1.1 presents an overview of how we used the three questions (RQ.1-RQ.3) to answer the main research question. Particularly, we start with exploring what kind of decisions GSD companies take in order to manage and control their multi-site activities. To that end, we propose a model for global software governance, i.e. a structured way to analyze governance decisions in GSD. We answer RQ.1 by combining results from background literature on GSD governance and results from case studies within companies, where we applied the proposed model.

Figure 1.1: Towards answering the main research question

Moreover, we found that different governance decisions steer collaboration between distributed team members in different ways. This observation motivated us to further explore the collaboration structures of GSD teams. For that purpose, we used social network analysis techniques to examine how members within a network connect with each other, what kind of characteristics bring members closer, and who are the most important members in the network. Synthesizing the results from a systematic literature review as well as our empirical observations,
we were able to answer the second research question (RQ.2).

Finally, multi-site collaboration restrains remote colleagues from knowing who knows what, who works on which part of the project and how to access and communicate with the right people. This observation motivated us to use transactive memory systems, in order to further study expertise and team awareness between geographically dispersed teams. To answer the last research question (RQ.3), we propose a two-fold approach in measuring transactive memory systems in GSD; First, a structural approach where we examine the collaboration structure of TMS using social network analysis. Second, a process approach where we build a latent variable model to measure the transactions/processes of the systems, based on existing literature on TMS.

1.5 Research methods

The research conducted in this thesis is both theory-oriented and practice-oriented. Theory-oriented research is used when the purpose is to explore the background literature and the concepts under study. Practice-oriented research is used when the purpose is to examine and analyze empirical data collected through, for example, interviews or surveys from industrial partners. Table 1.1 presents the research objective, research methods and data collection methods followed in every chapter, as well as the research questions answered.

The following paragraphs present the different research methods and data collection methods applied in this thesis:

1. **Case Study:** Case studies are the main research method followed in this thesis. Yin defines the case study as “an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident”. Based on the number of companies involved as well as the timeline of the research, three different types of case studies are applied:

   - **Single Case study:** when only one company participates in the research.
   - **Comparative Case study:** when data are collected and compared from two different companies.
   - **Longitudinal Single Case study:** when data from a single company are collected in two different points in time.
2. **Systematic Literature Review:** Systematic literature review is an evidence-based approach to thoroughly search studies relevant to some pre-defined research questions and critically select, evaluate and synthesize findings \[97\]. It is particularly powerful in collecting and analyzing existing work, which is a common task in establishing background knowledge in any research.

3. **Literature Review:** A literature review is less formal than a systematic literature review because it allows more freedom in collecting relevant studies and analyzing their content. Although the results of a literature review might not be as complete and valid as those of a systematic literature review, thanks to its effectiveness and efficiency, this method is often used to
gain certain knowledge or understanding on a specific topic.

In order to collect the data, we used several approaches. Interviews are the most common data collection method in this thesis. We performed semi-structured interviews with participants from the industrial partners. A semi-structured interview includes an interview protocol with specified questions, but at the same time it allows the researcher and the interviewee to further discuss other topics of interest [189]. A second data collection method that we extensively used is the survey, which is defined as the “collection of standardized information from a specific population or sample, by means of a questionnaire or interview” [160]. In this thesis, we use online questionnaires distributed to the participants through email invitations. Finally, in some case studies we used the approach of data triangulation, by combining the results of more than one data collection method. Data triangulation is broadly defined by [54] as “the combination of methodologies in the study of the same phenomenon”. The most important benefit of data triangulation is to tackle the internal threat to validity, by supporting results from one data collection method with results from another data collection. Additionally, data triangulation can capture a more complete, holistic, and contextual portrayal of the unit(s) under study [95].

1.6 Research projects and industrial partners

For the successful accomplishment of this research, we have collaborated with people from Océ Technologies and Ericsson. We also worked together with participants from OpeNER (Open polarity enhanced Named Entity Recognition), an EU-funded research project, under the seventh Framework Programme (FP7).

1.7 Chapters overview

Chapter 2 - Defining multi-site software development governance

This chapter outlines a multi-site software governance structure based on three aspects: the business strategy that binds the relationship of the remote offices, the structure and composition of the remote teams and the way tasks are allocated across sites. Knowledge management processes (including knowledge creation, knowledge transfer and communication) are identified and the influence of different governance structures on these processes is discussed. We do so through a case study at a multinational company in printing systems.
Chapter 3 - From RUP to Scrum: a case study

In this chapter, we apply the multi-site governance model to analyze the transition from a traditional development methodology (RUP) to an agile methodology (SCRUM). We identify the changes that the Scrum adoption brought, in comparison with the previous governance structure of RUP. We find that a transition from RUP to Scrum brings a positive effect in requirements engineering, communication, cost management and cross-functionality of the distributed teams. We also observe a negative change with regard to the development pace and delivery time.

Chapter 4 - Collaboration structures in global software development

This chapter presents different collaboration structures identified in GSD through social network analysis techniques. We perform a systematic literature review and we use the 3C collaboration model to classify our results based on communication, coordination and cooperation. Our results reveal two main coordination structures used in distributed teams, namely the clustering and the core-periphery structure. The analysis of the cooperation activities of the global networks reveal differences between planning and practice. Finally, several tools have been identified that aim to improve communication patterns among distributed team members.

Chapter 5 - Collaboration structures in software product lines

In this chapter, we present a case study where the software product line (SPL) organization changed from component-focused teams to product-focused teams. This goes against common advice, which favors a component-based SPL organization. We use social network data to analyze the collaboration networks of the SPL organization, before and after the change. We find that a component-focused organization of the teams leads to a centralized collaboration structure, while a product-focused organization results in a more de-centralized collaboration structure. Perceived advantages of this organizational transition involve a clearer focus and priorities for the teams, as well as less “ping-pong” communication. Some concerns for the future effectiveness of the new SPL organization include the effort to maintain a common platform across products, and the reduced specialization and expertise of team members.

Chapter 6 - Introducing transactive memory systems

In this chapter, we present the theory of transitive memory systems. Additional, we build a twofold approach for measuring TMS in global software development; A structural perspective, where we use SNA techniques to analyse the structure of TMS and a process perspective, where we measure the different processes (transactions) performed in a GSD environment, through a latent variable model.
CHAPTER 1. INTRODUCTION

Chapter 7 - The effect of governance on transactive memory systems

In this chapter, we present an analytical method to examine GSD governance decisions and their effect on TMS. We do so by collecting both qualitative and quantitative data from two multinational companies. Our results suggest that different governance decisions have a different impact on transactive memory systems. Offshore insourcing as a business strategy, for instance, creates tightly-connected clusters, which in turn leads to better developed transactive memory processes. We also find that within the composition and structure of GSD teams, there are boundary spanners (formal or informal) who have a better overview of the network’s activities and become central members within their network. An interesting mapping between task allocation and the composition of the network core suggests that the way tasks are allocated among distributed teams is an indicator of where expertise resides.

Chapter 8 - The role of brokers as facilitators of transactive memory systems

In this chapter, we examine the role of brokers as facilitators in the development of transactive memory. We use social network theory to analyze the collaboration of an EU-funded project, where development teams come from different partners and different locations. Our results suggest that task-based clusters emerge and that project members who coordinate activities as well as those who contribute to the code development act as brokers. Our empirical evaluation shows that clustering has a negative effect on transactive memory and that brokers can moderate that effect.

Chapter 9 - Transactive memory systems before and after software transfers

In this chapter, we present the results of a longitudinal case study in multi-site software governance. We examine the changes implemented over the past two years in the governance of an international company where software development activities and responsibilities were transferred between remote locations. We compare the governance model before and after the transfer. Particularly, we use the theory of transactive memory to evaluate the effect that the governance changes brought in expertise awareness, accessibility, credibility and communication frequency of the dispersed team members.

Chapter 10 - Conclusions

In this final chapter, we revisit the research questions introduced in chapter 1 and we summarize the answers provided throughout the thesis. We also reflect on the overall contributions of this thesis and conclude with suggestions of how the present thesis can be extended and further supported by future research.
1.8 Publications

Chapter 2 is published as:


Chapter 3 is published as:


Chapter 4 is published as:


The work in chapters 5 and 9 are submitted for reviewing as:


The work in chapters 6 and 7 is published as:


Chapter 8 was presented at the Eighth International Conference on Research Challenges in Information Science (RCIS’14) and was awarded as best paper. It is published as:

2

Defining Multi-site Software Development Governance

Software Development Governance (SDG) is an emerging field of research, under the umbrella of information technology governance. SDG challenges increase when software development activities are distributed across multiple locations. Coordination of knowledge management processes requires specific attention in multi-site development. This chapter outlines a multi-site software governance structure, based on three aspects: the business strategy that binds the relationship of the remote offices, the structure and composition of the remote teams and the way tasks are allocated across sites. Knowledge management processes (including knowledge creation, knowledge transfer and communication) are identified and the influence of different governance structures on these processes is discussed. We do so through a case study at Océ, a multi-national company in printing systems.

2.1 Introduction

Governance is defined as those arrangements and practices that an organization puts in place to ensure that the activities are appropriately managed [5]. In fact, only recently attention has been paid to define governance in software development: what are the structural attributes of software development governance and what are the coordination mechanisms that this governance embraces. SDG challenges increase when software development activities are distributed across multiple locations [157]. For example, Heeks et. al. [76] investigate different strategies in multi-site cooperation that can lead to either synching, that is a successful cooperation, or sinking, an unsuccessful cooperation.
CHAPTER 2. DEFINING MULTI-SITE SOFTWARE DEVELOPMENT GOVERNANCE

With the continuous and evolving strategies of distributed software development, scholars have been considering the challenges and the key issues of managing the knowledge in those distributed environments (e.g., [55], [176], [94]). For example, it was found that communication frequency and speed significantly drops among remote teams [81]. As a result, knowledge tends to “stick” to locations and it becomes difficult to transfer that knowledge from one site to another [175].

In this chapter, we argue that the way software companies govern their multi-site development activities, can impact the management of knowledge across remote locations. We investigate two different types of multi-site software governance through a case study, in a multi-national organization. We identify several knowledge management challenges that emerge in these distributed environments, and we observe how the governance structures impact these knowledge management challenges.

2.2 Governance in global software development

Software Development Governance (SDG) is an emerging field of research, under the umbrella of information technology governance. The goal of SDG is to ensure that business processes of the software company meet the strategic requirements of the organization [110]. In order to achieve this, software development processes should be aligned with the company’s business goals on a strategic, tactical and operational level. Chulani et. al. [33], for instance, describe a unified view of SDG from the start at the senior executive level and all the way down to the practitioner level where projects are implemented. Governance assigns decisions within the organization, whereas project management implements those decisions [33].

The challenges of SDG increase, when software development activities are engaged in a multi-site environment. Several proposals have been made to define the attributes of a software governance model for distributed development projects, and the coordination mechanisms that this model should embrace. Ramasubbu and Balan [157] have proposed a research model on how to create a governance framework for distributed software development, focusing on three stages of a project lifecycle; planning, execution and reflection. Gewald and Helbig [71] suggest a governance framework for managing outsourcing engagements based on organizational structures, joint processes and relationship management functions.

As Bannerman [5] suggests that SDG should include both a functional perspective in terms of what governance does as well as a structural perspective dealing with
2.2. GOVERNANCE IN GLOBAL SOFTWARE DEVELOPMENT

*what governance looks like.* Based on this, Chulani et. al [33] define software development governance as:

- Establishing chains of responsibility, authority and communication to empower people within a software development organization (structural component of governance).

- Establishing measurement and control mechanisms to enable software developers, project managers and others within a software development organization to carry out their roles and responsibilities (dynamic or functional component of governance).

In this paper, we make our contribution and define a multi-site software development governance model from a structural perspective. We outline a multi-site software governance structure, based on three aspects: the business strategy that binds the mutual relationship of the remote offices, the structure and composition of the remote teams and the way tasks are allocated across sites. In the rest of this section, we elaborate more on each of those attributes and the reasons why we define them in the context of multi-site software governance.

2.2.1 Business strategy

According to Gewald & Helbig [71], a multi-site governance model should steer and control the cooperation of the remote offices, based on partnership and mutual trust. Carmel and Tija [26] note that one of the things that companies should not forget when they operate in a global environment is the broader strategic goals and their legal implications. Additionally, organization design mechanisms define patterns of dependence and cooperation [102]. Multi-site business strategy should therefore consider the contractual and legal relationships between the remote sites and the implications these might have in their collaboration, communication and knowledge management process.

2.2.2 Team structure and composition

In software development, team structure and composition is a critical factor for good performance [138]. Team size, role descriptions and role distribution are among those characteristics in distributed teams that can influence team coordination and communication and therefore team performance (e.g. [81], [142], [63], [143]). For example, Kofman and Klinger [99] suggest that role confusions (meaning the confusion caused by the difference between how the role is described and what people actually do within the role) may affect team performance and
communication. Faraj and Sproull [65] also propose expertise coordination of a team - that is the coordination of knowledge and skill dependencies - as an important component of teamwork coordination for knowledge teams. In this paper we argue that team structure and composition is an integral part of multi-site software development and it should be considered part of the governance structure.

### 2.2.3 Task allocation

Several criteria exist on how to distribute work across sites, such as based on the area of expertise, on the software architecture and structure, or even based on the development process steps from requirements elicitation and communication to maintenance [106]. Work dispersion and the interdependencies between the distributed tasks influence the coordination and administration of the remote offices [81]. For example, Mockus and Weiss [128] argue that when assigning a task to a remote site, it is important to consider that the allocated task matches the development-resource capacities of that location. Additionally, according to Carlile [24], as the number of dependencies between actors increases, the complexity and the amount of effort required to share and access knowledge at a boundary also increases. Hence, we consider task allocation as a fundamental attribute of multi-site software governance.

### 2.3 Knowledge management in global software development

Software development is a knowledge-intensive process and knowledge management challenges increase when the development activities are distributed across multiple locations. Managing the knowledge in multi-site software development includes the coordination of communication between the remote teams [2]. Distance, for instance, between the teams introduces barriers to informal and face-to-face communication, and the collaboration of the remote colleagues is dependent on synchronous or asynchronous communication tools [48]. Additionally, communication speed and frequency is influenced by the coupling and the interdependencies among the distributed tasks. It is suggested for example, that tightly-coupled tasks that are distributed across multiple sites, can increase the communication frequency [128].

Another knowledge management challenge in distributed software development is the knowledge capture which is seen as the process of recording knowledge in...
2.4. PROJECT OVERVIEW

a medium, that is transforming and encoding it as information. According to Correia and Aguiar [36], the effectiveness of how knowledge is captured into artifacts, and acquired by other team members, is of crucial importance to a project’s success. Research also suggests that in distributed software development, documentation must be current and reflect what various teams are using and working on, to prevent assumptions, misunderstandings and to support maintainability [82].

Moreover, several factors can impede knowledge transferability across sites such as the use of different working methods or the differences in skills and expertise of the remote colleagues [144]. For example, Carlile [24] argues that as the differences in the amount and the type of knowledge that people possess increases, the effort required to share and transfer that knowledge also increases. Furthermore, transactive memory - the knowledge of who knows what and where knowledge resides [182] - has been studied as a supporting structure in knowledge transfer across multiple locations. Kotlarsky and Oshri [100], for instance, investigated two globally distributed system development projects and argue that transactive memory as a mean of knowledge sharing contribute to successful collaboration across remote teams.

In this paper, we investigate knowledge management challenges that can emerge in a multi-site software development environment. Based on the aforementioned literature we classify the identified challenges under three main categories, communication, knowledge creation and storage and knowledge transfer. We also make a distinction between system-generic and unit-specific knowledge. We define system-generic knowledge as the comprehensive knowledge of the entire system that teams are working on. In other words, it is the knowledge of how the end product looks and functions. Unit-specific knowledge is the particular knowledge that the individual has, for the specific unit he or she is working on.

2.4 Project overview

The research was conducted at Océ [1], a multi-national company in printing systems, part of the Canon Group. Océ is headquartered in the Netherlands, with offices in more than 100 countries and over 20,000 employees. Research and development departments work on hardware as well as software innovations, and are located in nine different countries.

Océ successfully applies an agile development methodology to encourage creativity and productivity. The organization is flat and employees are encouraged to

1www.oce.com
be proactive in owning up to work responsibilities. Océ has deliberately opted for cooperation, as opposed to hierarchy, to foster innovation and entrepreneurship. People are not curtailed by strict processes and the drive to deliver business results is strong, so the latter will take precedence over writing excessive documentation, and a lot of knowledge therefore remains transactive.

This case study focuses on one business unit which specializes in high-end printers. The development of the software used in those printers is distributed among the main site (site NL) and two remote sites: site A and site B. The software includes units such as accepting requests, controlling print jobs, rendering images, controlling devices (e.g. scanners) as well as local and remote user interfaces.

In the next subsections, we first sketch a typical team structure at Océ followed by the business strategy, team structure, and task allocation in the site NL - site A and site NL - site B coalition, respectively.

2.4.1 Project team structure

In a typical team structure of project teams at Océ, a project manager heads the project and is responsible for its planning, realization, and successful completion. The project manager also agrees upon the high-level specifications of the project with upper management and marketing personnel. Requirements and specifications are compiled into product properties by the lead architect in the team. The specifications written by the lead architect start from a user-centric view, i.e., scenarios on how the end-users will interact with the product. For instance, the architect is responsible to decide what happens in case of a request from the user of the printer as in how should it function, which software units should be triggered and how should they function, define the interfaces between these units and the like. Teams also comprise a system integrator who integrates the different software units to build the software system. Additionally, the system integrator reports issues encountered during integration and assigns them to the appropriate team or person to be addressed. All three – the project manager, the lead architect, and the system integrator – are assigned to a project for its entire duration until the product has been released.

Project teams also comprise one or more software unit teams that implement the software units. Each unit has a unit leader and a unit architect analogous to the project manager and lead architect at the project level. The unit leader is responsible for planning and organizing. The unit architects transform the high-level specifications received from the lead architect into detailed technical specifications and pass them to the software engineers who implement the code and test it. The unit architects are coordinated by the lead architect, who is
often the only team member with the overall view of where (or in which units) do the different functionalities of the product reside. Most software units are not developed for a single product but their deliverable is tuned and integrated into several products. A software team can develop a software unit for four, or even more, projects at the same time. This challenges system behavior as well as architecture.

2.4.2 Relationships between site NL and site A

The collaboration between site A and site NL concerns the co-development of a software unit, which means that the development activities of that unit are distributed among members in a team in site NL and members in a team in site A. Not all information available at site NL can be freely shared with site A.

For the unit developed between site A and site NL, the unit leader is located in site NL. In site A, another role is created, referred to as the team leader, who is the coordinator of the local team at site A as well as the main contact person with the team in site NL. Both teams consist of software engineers, testers as well as unit architects.

Development tasks are allocated to the software engineers of the two sites by the unit leader of the specific software unit. Two main criteria are applied in deciding which tasks should be sent to site A: the first is based on previous experience, i.e., whether a software engineer has worked on a certain task before, and therefore he or she is more capable in dealing with a specific change request, resolving related defects or adding new features. The second criterion is the complexity of the task. If requirements are difficult to communicate through email, or via the phone, the tasks are more likely to be assigned to someone at site NL.

2.4.3 Relationships between site NL and site B

Site B is responsible for the development of a software unit, with an independent unit team that develops the unit and ships it back to site NL for integration. The team in site B has a unit leader, a unit architect and software engineers and testers. All information available at site NL can be shared with site B.

In the beginning of each release cycle, the project manager from site NL and the unit leader from site B, as well as the lead architect from site NL and the unit architect from site B, create a plan for the next release and discuss the requirements to be implemented. At the end of each release, the work developed
and tested in site B is shipped back to site NL for integration and system and product testing.

### 2.5 Research methodology

The research was conducted based on a qualitative data analysis approach. Qualitative research refers mainly to the investigation and analysis of personal experiences and behaviors, as well as organizational functions and social interactions [174]. In order to gather the required data for the analysis, we chose to perform semi-structured interviews. In semi-structured interviews, questions can be open-ended allowing a conversational manner, while at the same time, an interview protocol can still be followed [189]. We also had several focus groups with contact people from the company, during which we discussed details of our research, presented our findings and interpreted the results.

To gather the data, 20 interviews were conducted with a duration of approximately 90 minutes each. An interview protocol was designed to guide the discussions, which covered questions on the topics of communication and knowledge management. The respondents included employees from all three sites, with different roles and positions. Table 7.2 presents the number of the respondents, their roles and the location. Site A has no system integrators, unit architects or designers, therefore no interviews from these roles at that location were held.

<table>
<thead>
<tr>
<th>Table 2.1: Interview Participants across locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site NL</td>
</tr>
<tr>
<td>Software Engineers, Testers, Unit Architects</td>
</tr>
<tr>
<td>System Integrators, Unit Architects, Designers</td>
</tr>
<tr>
<td>Project Managers, Team Leaders</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

All interviews were recorded and transcribed. To process the data, the Atlas.ti tool was used, a commercial software for qualitative analysis of textual and visual data. The interviews were analyzed based on the coding process of microanalysis or otherwise called a “line-by-line” analysis [174]. The codes were eventually grouped into emerging concepts. These concepts reflect the identified knowledge management challenges and the multi-site governance structures that impact those challenges. In a companion article [34], a larger set of interviews
is used to validate a collection of software architecture knowledge management practices.

## 2.6 Knowledge management challenges

### 2.6.1 Communication

Within the case study company, several observations can be made with regard to communication. Between site NL and site A, communication frequency is more intense among all the team members and across all development phases, from software engineers to unit leaders. Between site NL and site B communication frequency depends on the development phase. More specifically, a more intense communication seems to be necessary in the beginning of the release cycle during requirements planning and communication, as well as near the end of a release cycle during integration and testing. Figure 2.1 qualitatively illustrates the communication frequency of site NL - site A and site NL - site B coalition.

![Figure 2.1: Communication Frequency through the development life cycle](image)

In a distributed development environment, communication can be either asynchronous using emails and chats, or synchronous through video conferences and face-to-face meetings. Synchronous communication is generally considered preferable in distributed environments. The reason is that a small issue can take longer to be resolved using emails circulating between sites, while a brief conversation through the phone can quickly resolve and clarify issues. Communication
speed and frequency is important in multi-site environment, and potential delays in communication can cause delivery and project delays.

Based on the need for communication among the different roles and across locations, the use of the different means of communication varies. Software engineers and architects use an instant messaging (IM) tool on an everyday basis. The IM tool allows for direct exchange of messages as well as the possibility for desktop sharing. This synchronous communication part of the chat tool covers the need for the everyday communication that software engineers and architects have. The introduction of the instant messaging (IM) tool was a significant improvement in the communication between site A and site NL, during the last year. Through the IM tool, they can see when some of their remote colleagues are online and available at the moment, and they come directly in contact with the person they need. Considering the high needs for frequent communication that the two locations have, the addition of a complementary tool of synchronous communication was perceived from the employees as a great advancement in the collaboration methods.

“The advantage of chat is that it is much faster. Once the person is available, you start asking and you can do this very fast in a short time” (Team leader, site A)

With the IM tool, software engineers can also use desktop sharing, improving communication speed and potentially improving performance. Desktop sharing permits remote colleagues to have a real time, interactive communication while they share their desktop environments. They can check each other’s work, if certain features have been implemented the right way, or solve bugs by tackling problems together and assisting each other.

“And we can also talk and give control and then you can go sort of around the code and check and it’s very fast. Desktop sharing covers pretty much all the needs I have.” (Software engineer, site A)

“We share the desktop of a developer - like this morning we did it again to just get a feedback on the look and feel of an implemented part or behavior - and then you can, together with the developers in site A and site NL, you can click through stuff and see what has been implemented before and check in the code.” (Unit leader, site NL)

In the case of unit leaders, phone calls and organized video conference are the most common means of synchronous communication. The difference is that unit leaders, although they still need to communicate on an everyday basis, hardly ever work all day in front of their desk. Chatting therefore cannot serve their needs, and phone calls are preferred. Email seems to be the next favorite tool for
communication for all roles. It serves best when many people need to be involved in a discussion, or informed about a decision. At the same time, email allows for a more direct means of sharing documents with many colleagues. Finally, traveling is more common for unit leaders, and less so for software engineers.

2.6.2 Knowledge creation and storage

According to Hansen et. al. [73], codification is pursued when knowledge is documented and stored in databases, while personalization relies on the tacit knowledge that people possess and sharing that knowledge in person-to-person communications. With their choice for an agile development method, site NL has deliberately opted for a strong personalization strategy. Employees are encouraged to cooperate with each other, taking initiative and responsibility without being constrained by strictly defined processes. With this approach, much of the knowledge remains tacit and details will not always be updated in the corresponding documents.

“And sometimes there are documents, the system behavior documents, and system architecture documents, but they are mostly on a higher level.” (Software Engineer, site NL)

“Most of the time you don’t just look for information in documents or in databases, but you walk to a person whom you know is working on stuff or knowledgeable and you just ask questions. And maybe that person also doesn’t know but he will give you a pointer to somebody else, and so you just do it by communicating and do it by talking to people.” (Unit Leader, site NL)

The observed codification and personalization practices within site NL have certain implications for site A and site B. In the case of site B, we noticed that documents are stored and shared with site NL through the use of different repositories and tools. These tools include development platforms and applications as well as communication tools like emails and chats. Consequently, people find it difficult to search through all these databases and repositories to locate the right document and the up-to-date information they need. Because of this situation and because of the relative development independence site B has, some local codification strategies have developed and site B maintains its own internal way of documenting knowledge. This independence, but also separation, from the working methods of site NL, inevitably creates a gap in knowledge sharing between site NL and site B.

“We have a SharePoint where we put a lot of the documentation;
that’s a recent technology we use here. I think it’s been introduced over the last year. And it’s convenient for all people because previously that was quite a mess, because you have one version of the document in one share folder on the network, and the other, the same document elsewhere.” (Software Engineer, site B)

Documentation storing and sharing between site NL and site A also has certain characteristics. Because of the information barriers between them, the team at site A does not have access to all the documentation available in the databases of site NL. As a consequence, site A employees need to ask their colleagues from site NL for any extra information they might need. In turn, site NL people need to put extra effort and time to find that information and send it back to site A. In Océ, proactiveness is encouraged, but so is asking for information one needs to do a proper job.

“I don’t mind. I’m actually happy that they are asking for information, because we’ve seen a lot of cases where they just didn’t ask and they did something or assumed something, and afterwards they complained they didn’t have anything - they didn’t ask! We’re really in a mode that people should ask for information, so I’m really happy they ask, so then already I quickly see what it’s about and see if I can help” (Software Engineer, site NL)

2.6.3 Knowledge transfer

The majority of the project is developed at site NL and therefore most of the system-generic knowledge also resides in site NL. This inevitably causes knowledge to be “sticky” \(^{175}\), which means that it takes additional effort for that knowledge to be transferred from site NL to the remote sites.

“We don’t feel the same handicaps as our remote colleagues, and maybe, I hope I do my best to keep them in touch and things like that, but yes, here we can act much faster.” (System Integrator, site NL)

Another reason that enforces the stickiness of knowledge is the lack of the actual physical printing machines at the remote sites. The teams in the remote sites develop and test their work using simulators or they have a type of printer with limited functionality. This restricts their knowledge on how the entire system works, and their capabilities to perform as good as their colleagues in site NL.

“But I do think that having a machine would help in understanding how the system works and, for now I don’t understand yet why they don’t have a machine.” (Unit leader, site NL)
Concerning the unit-specific knowledge, the situation seems to differ between the co-development cooperation site NL - site A and the independent development site NL - site B. In the first case, function-specific knowledge remains in site NL and it is difficult to be transferred to the site A team. The main reason appears to be the lack of the printing machines in the remote site that limits their capabilities, as well as the unit-specific knowledge of the team in site A.

“This year it was the first time that we saw also how this kind of testing is done. It was all taped and they made a sort summary and we just saw how something was tested and so on. But it was the first time we saw that. And then we heard the customer saying, yes, we’d like this and that and then we heard, yes, we will have to do that and that and you will get the requests and so on.” (Tester, site A)

Another challenge in knowledge transfer is how to locate the knowledge. As we have previously observed, a strong personalization strategy is pursued in agile development methodologies. People rely more on the transfer of knowledge in a person-to-person way, and it becomes more efficient to know who knows what. Transactive memory is highly visible in the agile environment of site NL and it has been recognized as an efficient way of improving performance, since people can find the right information by spending the least effort and time.

“And it makes it easier to know who is working on what part of the implementation, and that’s also what we learn by being there, that you are more aware of what question should I ask and to whom. Don’t just blow out your question and send it to five engineers, but when we talk about this part of the end phase, I know I have to contact this person” (Designer, site NL)

### 2.7 Multi-site software governance

#### 2.7.1 Business strategy

The contractual relationships and as a result the information barriers between site NL and site A seem to impact knowledge sharing between remote colleagues. People in site A do not have direct access to the documentation that is available in site NL, and as a consequence they rely on the selected information and documentation that site NL sends them.

“Site A is some kind of different entity. And therefore, we always have to ask, please, or we don’t have access to all the repository that
they have and usually then we have to ask, please, could you please

copy that from there to here so that we can have also access and

sometimes it might be annoying but . . .” (Software Engineer, site A)

Additionally, people in site NL and especially the software engineers have to

invest more time and effort to filter the information and share it with their remote

colleagues.

“So they cannot access the Wiki, so in the beginning we put a lot

of stuff in the Wiki and they couldn’t access it, so problem. Later they

said, well, we could use sharepoint and put stuff on there, and you have

versioning so it’s kind of a mix between Wiki and storing documents

and, well, I’ve actually never really used it a lot.” (Software Engineer,

site NL)

Finally, the flat organization and the agile environment in site NL enforce the

quick response time from people working in that site. Software engineers in

site NL are encouraged to take responsibilities and initiative and act accordingly

and they can quickly reply to their colleagues in site A, increasing the speed of

communication between the two remote locations.

“I expect that people are perhaps in meetings and they are quite

busy, but usually they are able to answer me right away and it’s very, very good for us.” (Software engineer, site A)

2.7.2 Team structure and composition

In distributed teams the lack of team cohesion is a recognized challenge [158]. Team members in remote locations are less likely to perceive themselves as part of the same team, compared to team members working in a co-located environment. Moreover, in a highly agile environment, there is less attention to documentation and as a result much knowledge remains tacit. Transferring tacit knowledge requires personal interactions and in a multi-site environment these interactions need more effort and time to take place.

As described earlier, the largest part of the project remains in site NL. Consequently, more people are working there and more knowledge resides at that site. Most of the software units are developed in site NL, and most of the software unit-specific decisions are taken there. Additionally, the unit leaders and system integrators are located in site NL which means that also all the system-generic decisions are taken in site NL. Since most of the project knowledge is created in site NL and remains agile, people are less likely to understand the need to transfer that knowledge to their remote colleagues.
2.7. MULTI-SITE SOFTWARE GOVERNANCE

“We are big brother, and they are only small team, and yes, we can settle a lot of issues, at the coffee machine, and yes, they don’t participate. But that’s the organization. If we were with three equally groups, one in site A with their responsibility, one in site B with same responsibility, and one in site NL with same responsibility, that would be different. We also would feel that we can’t just decide something at the coffee machine, we have to put it on paper and we have to agree before the parties, but since we have the system architect and the biggest role, its too easy not to document things, too easy not to have a meeting about it, I think that’s the case for a large part.” (Unit leader, site NL)

The hierarchical organization of the teams might influence knowledge sharing. Previous research suggests that as the role differentiation increases in software development teams, it leads to a decrease in interaction and a corresponding decline in the shared mental model [111]. Within Océ, it is perceived that the hierarchical organization is more intense among the team structures in site B. By hierarchy, we mean that a unit leader for example has more decision making power on design and development matters than a developer. As a result, developers in site NL believe that their colleagues in site B need more time to take an action (e.g. fix a bug or work on a Change Request) and that often can cause delays; this has not been quantitatively confirmed though.

“I had to wait for days through the official channel of delivering, of updating.” (Software Engineer, site B)

“Usually I’m telling him I discussed this or if somebody is requesting to do something first we have to ask here on our site, is it okay for you if I do this because I was requested, and if he says, yes, then we are doing it. If he says, no, we have to say, sorry, can’t do it yet.” (Software Engineer, site A)

Finally, team structure and composition can impact the communication speed and frequency. Based on the way teams are organized within the company, interaction between remote locations is more likely to occur through the “contact persons”. This can potentially create communication bottlenecks and delays.

“It’s better if you inform your responsible before you contact. But it might depend on responsible, also. There are some that they need to know everything before you contact the remote location, and sometimes you have a responsible that delegates to you the responsibility of something.” (Software Engineer, site A)
2.7.3 Task allocation

In task allocation, we examine the way development activities are distributed among sites. We have already mentioned the two main task allocation methods identified in the Océ case study (co-development between site NL- site A and more independent development between site NL - site B). During the analysis, several knowledge challenges appear to be influenced by this construction. An important aspect is that, most often, the task allocation is tightly coupled with the team structure and composition [105], and more specifically in distributed development environments [6].

One of the first things to be noticed is the influence of task allocation on the communication frequency. As already described previously, communication frequency between site NL and site A is higher. On the other hand, communication frequency between site NL and site B depends more on the development phase. Consequently, we observe that tightly-coupled activities require a more intense communication compared to loosely-coupled activities. Team composition also plays a role in this case, as the fluctuation in the communication frequency refers primarily to software engineers and unit architects. Unit leaders who communicate for progress status and planning purposes have a more stable communication frequency based on scheduled regular meetings. For example, there is a Software Progress Committee that meets once every week to discuss project planning and current status.

In addition, the codification strategy appears to be influenced by task allocation. In the case of co-development between site NL and site A, where development activities are tightly coupled, it is also noticeable that remote colleagues are more dependent on documentation sharing. Site B, however, has a self-sufficient domain knowledge and their communication with site NL is on interfacing and integration. There is a local Sharepoint site within site B and the team has its own, local procedures of how and when to document and share knowledge, limiting their codification dependencies with site NL.

The location where domain knowledge resides, is also connected with task allocation. In co-development, all knowledge (system-generic and unit-specific) remains in site NL and therefore additional effort needs to be invested in knowledge sharing between the remote sites.

Finally, task allocation can have a social impact on knowledge management. The motivation of site A colleagues can be influenced by the lack of responsibility in the development tasks that they receive, and they more often leave the company. When an employee leaves the company that means that he or she takes the knowledge obtained so far, and at the same time Océ has to hire someone else,
2.8. LESSONS LEARNED

train him and integrate him with the team, the project and the company.

“You just get small pieces of the cake. This is also frustrating for the guys here, because they are, in terms of qualification, they know the technology, at least the same level as the guys in site NL, but they lack the top level domain knowledge, and this causes instability in the team, because one could say, okay, I’m very good in Java. I can do more than this small piece of user interface.” (Team leader, site A)

2.8 Lessons learned

We have examined two multi-site software governance structures and their differences. The first difference relates to the business strategy that binds the relationship between the remote offices. Between site NL and site A there are information barriers. The next difference is the way tasks are allocated. The site NL and site A teams co-develop a software unit, which implies that the development activities and dependencies between the two sites are tightly coupled. On the other hand, site B is responsible for the development of a complete software unit, meaning that the development activities are more independent from site NL. Finally, the two governance structures are similar as far as team structure and team composition is concerned. Site NL is a flat organization, with small distances between roles, and agility is highly supported. The remote sites however are more hierarchically structured and focus more on strict processes and procedures.

Having defined the two cases of multi-site software governance, we investigated the impact different governance structures might have on several knowledge management challenges. Table 2.2 presents our main findings. The knowledge management challenges identified within the case study concern the communication, the knowledge creation and storage and the knowledge transfer. The main impacts observed are:

- The information barriers between offices increase the effort and time spent on managing the creation, storage and sharing of knowledge, both tacit and explicit.

- The unbalanced structure and composition of teams impedes the smooth flow of knowledge. For example, when the team leader of the unit is located in site NL, unit-specific knowledge “sticks” to site NL.

- Allocating tightly-coupled activities among remote teams, increases the need for knowledge sharing and more effort needs to be spent in knowledge
CHAPTER 2. DEFINING MULTI-SITE SOFTWARE DEVELOPMENT GOVERNANCE

Table 2.2: The impact of Multi-site Software Governance on Knowledge Management

<table>
<thead>
<tr>
<th>Multi-site Software Governance</th>
<th>Knowledge Management Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Strategy</strong></td>
<td></td>
</tr>
<tr>
<td>Site NL-Site A:</td>
<td>• No direct documentation due to information barriers.</td>
</tr>
<tr>
<td>• They are different companies and information barriers exist between the remote sites</td>
<td>• Information sent from Site NL to Site A needs to be filtered.</td>
</tr>
<tr>
<td>• Communication frequency is higher.</td>
<td></td>
</tr>
<tr>
<td>Site NL-Site B:</td>
<td></td>
</tr>
<tr>
<td>• They are the same company and no information barriers exist between the remote sites</td>
<td></td>
</tr>
<tr>
<td><strong>Team Structure &amp; Composition</strong></td>
<td></td>
</tr>
<tr>
<td>Site NL-Site A:</td>
<td>• Hierarchical structures create bottlenecks in knowledge sharing.</td>
</tr>
<tr>
<td>• Site NL is a flat organization, while Site A is hierarchically structured.</td>
<td>• Too much focus on agility stresses tacit communication and documentation remains outdated.</td>
</tr>
<tr>
<td>• Role descriptions differ between sites.</td>
<td>• Different role descriptions makes knowledge difficult to locate.</td>
</tr>
<tr>
<td>• Unbalanced team sizes.</td>
<td>• Knowledge tends to stick where the majority of teams, or where the larger teams are located.</td>
</tr>
<tr>
<td>Site NL-Site B:</td>
<td></td>
</tr>
<tr>
<td>• Site NL is a flat organization, while Site B is hierarchically structured.</td>
<td></td>
</tr>
<tr>
<td>• Role descriptions differ between sites.</td>
<td></td>
</tr>
<tr>
<td>• Unbalanced team sizes.</td>
<td></td>
</tr>
<tr>
<td><strong>Task Allocation</strong></td>
<td></td>
</tr>
<tr>
<td>Site NL-Site A:</td>
<td>• Tightly coupled activities increase the need for knowledge sharing.</td>
</tr>
<tr>
<td>• They co-develop a function and their activities are tightly coupled.</td>
<td>• Co-development creates a greater need for codified knowledge.</td>
</tr>
<tr>
<td>• Communication frequency is high.</td>
<td></td>
</tr>
<tr>
<td>Site NL-Site B:</td>
<td>• Knowledge tends to stick to the independent development teams.</td>
</tr>
<tr>
<td>• They develop independently and their activities are loosely coupled.</td>
<td>• Communication frequency depends on the release phase.</td>
</tr>
</tbody>
</table>

transfer. The communication frequency is higher because team members need to be in contact on an everyday basis for the coordination of their development activities.

Summarizing our observations, the software governance between site NL and site A has a stronger impact on knowledge management than the software governance between site NL and site B. Whether the observed impact is beneficial for the productivity and the performance of the distributed project—for example in terms of development speed or number of residual bugs—needs further study. We round off this research with a structural perspective of multi-site software governance associated with (a) the business strategy between the remote offices, (b)
the structure and composition of the distributed teams, and (c) the allocation
of the development tasks across sites. We believe that these attributes should
be considered for the success of a multi-site software governance structure. We
should not forget, however, that there is no “one size fits all” solution to gov-
ernance but that an effective governance is rather dependent on the situational
characteristics of a software company [5].

2.9 Conclusion

We have defined a multi-site governance structure based on business strategy,
team structure and composition and task allocation. We also identified several
knowledge management processes, within the distributed software development
environment, dealing with communication, knowledge creation, storage and shar-
ing. In our case study, information barriers between sites, unbalanced team
composition and tight coupling of distributed activities impact knowledge man-
agement processes.

Multi-site software governance should be further researched and enriched, by
eliciting more information and identify other governance aspects besides we iden-
tified so far. By creating a more concrete definition of what constitutes multi-site
software governance, we can build more accurate models and perform additional
case studies on how to organize and administer multi-site software development
activities at a strategic, tactical as well as operational level. Our aim is to ex-
pand and supplement the results of the present case study and create a multi-site
governance framework based on best practices. This way we obtain a sound in-
sight into what steps should be taken, and how organization and development
activities should be structured to best align business and software development
goals.
From RUP to SCRUM: A case study

In this chapter we present the results of a case study at two offshore projects that recently adopted the agile way of working. We analyze their multi-site governance activities adopted and adjusted based on the Scrum methodology. Furthermore, we identify those changes that the Scrum adoption brought, in comparison with the previous governance structure of the Rational Unified Process (RUP). We find that a transition from RUP to Scrum brings a positive effect in requirements engineering, communication, cost management and cross-functionality of the distributed teams. We also observe a negative change with regard to the development pace and delivery time. Overall, we add to the body of knowledge in the field of distributed agile, with an additional field study where we describe and compare the migration from RUP to Scrum, and the implications of this transition.

3.1 Introduction

In Global Software Development (GSD), agile methods are gaining in popularity [158]. Agile methods can help improve communication and collaboration in offshore development, which often results in better business / IT alignment and responsiveness to business changes [130]. However, when using agile methods in GSD, the main principles of agile become more challenging: direct face-to-face communication occurs less often and the team is not collocated anymore. In offshore development, communication and collaboration is always more challenging [1]. Since agile relies more on good communication and face-to-face contact than other software development methodologies, the implications will be far higher as well.

Whether a global software company decides to adopt an agile way of working or
more traditional methodologies, it must be clear how the multi-site activities are governed. In other words, in order to organize and manage their distributed software development activities, organizations must adopt a governance model [76]. The selection of certain governance activities can influence the communication and knowledge management activities between the distributed teams [121].

In this chapter, we present the results of a case study at two offshore projects that recently adopted the agile way of working. The projects are developed in a consultancy firm located in the Netherlands, together with an offshore site in India, and the client is a finance firm. We use the multi-site governance model introduced in Chapter 2, to analyze their multi-site governance activities adopted and adjusted based on the Scrum methodology. Furthermore, we identify those changes that the Scrum adoption brought, in comparison with the previous governance structure of the Rational Unified Process (RUP). The aim of the study is to add to the body of knowledge in the field of distributed agile, with a complementary field study where we describe and compare the migration from RUP to Scrum, and the implications of this transition.

3.2 Distributed agile: benefits and challenges

The use of agile methodologies in global software development is an emerging trend. Usually, organizations turn either to offshore development or to agile development to deal with software engineering challenges like reducing development costs, improving quality, aligning business and IT. Agile aspects like intensive collaboration and low-weight documentation seem to make the two approaches diametrically opposed. Intensive collaboration is quite challenging when one part of the team is located at an offshore location. Distributed software development often requires more documentation and a strict plan. Yet, it seems organizations who successfully overcome the complexities of merging the two approaches can gain advantages from both.

The transition from the more traditional methodologies towards an agile way of working is a challenge that comes with both benefits as well as risks. Nevertheless, such a change must been seen as an “ally rather than an enemy” [10]. The challenges that can accompany a transition from the traditional methodologies to an agile approach may relate to the requirements engineering, communication, cost estimation and other process-related challenges. Global software development is usually characterized by engagements with different national and organizational cultures in various geographic locations and time zones. Therefore, GSD practitioners need to employ suitable context specific mechanisms to mitigate these problems [87].
3.2. DISTRIBUTED AGILE: BENEFITS AND CHALLENGES

An important part of global software activities that can be influenced by agility is the requirements related activities such as requirements elicitation, prioritization, design and communication \[107\]. Requirements engineering is a difficult task even in a co-located environment, but it becomes even more challenging when requirements must be managed across time, distance, cultural and language differences. A prominent issue for example that prohibits requirements elicitation in GSD is the distance between the client site and the development team, as well as the trust issues that might exist between the two parties \[49\]. Agility seems to help mitigate challenges in requirements engineering in global environments. For example, agile methodologies such as the XP programming, promote a more frequent interaction between the customer and the developer which ultimately brings the two parties closer to each other and “motivates” the distributed teams to improve communication volume and frequency between them \[49\]. Furthermore, frequent integration and testing also help to ensure that every team member has understood the requirements correctly \[145\]. This is especially helpful if the participants in a globally dispersed team are from different cultures and have not worked together before. Through frequent integration and testing, team members will get a lot of feedback and any misunderstandings become visible in an early stage of the project. This prevents problems to grow or accumulate \[145\]. Short iterations bring also transparency of the work progress to all participants. Offshore developers for example can get instant feedback on their work which helps to motivate them and build trust between the different participants in the project.

Agile methods emphasize communication and provide useful coordination and collaboration practices. Applying those practices in offshore development can help improve many issues related to distance \[85\]. Increased collaboration and communication between the dispersed team members can also reduce cultural differences between the participants \[145\]. Frequent and open communication between team members and the frequent releases to the customer builds trust and helps to better understand each other’s culture. Within the field of knowledge management, intensive collaboration and communication can increase shared tacit interpersonal knowledge between the distributed team members and reduces the need of sharing explicit knowledge \[1\]. For example, it can improve the knowledge on the business domain of the customer at the offshore site and it lowers the need to make these requirements clear through extensive documentation.

Applying agile methods in offshore development brings also additional flexibility to the contractual relationship between the involved parties. In the traditional global software projects, due to the limited ability to control the activities of the distributed teams, projects often rely on fixed commitments and pre-defined requirements \[158\]. On the contrary, agile processes are more flexible and adaptive...
and they emphasize changes during the development process. The customer can apply changes during the development phase without big consequences related to contract negotiations [145]. The customer does not need to specify all the requirements beforehand since agile processes are adaptive and focus on creating value to the customer. Requirements can change over time and they can be derived from a constant negotiation between the developer and the customer.

Finally, agile methodologies use short iteration, frequent builds and continuous integration. These practices bring challenges to configuration management and version control in software engineering in general [145]. This practice is even more challenging in global software development where part of the projects must be managed over different, distributed locations. When teams are dispersed, long distances and bad infrastructure can create extra impediments in communication and cooperation.

The status of combining agility and GSD is captured in two systematic reviews [86, 93]. Many of the case studies reported therein concern situations where one company is developing software for its own use, or a product they sell, and outsource part of the development to another party. That is, there is no third party client. Examples hereof are [179] and [139]. If there is an outside client, he is not part of the team, as in [146]. Another typical aspect is that many of the case studies concern situations where the different sites operate relatively independently, and their work is coordinated at regular intervals. In the case of Scrum, this means different Scrum teams at the different sites, and Scrum masters that coordinate the work through a Scrum of Scrums, a practice advocated by the Scrum Alliance. This is different from a fully distributed Scrum team, where members of one Scrum team are distributed across geographies. An example of the latter is [146].

### 3.3 Project overview

The case study was conducted in a software consultancy firm, located in the Netherlands. We focus on two development projects that the company was working on, for a financial firm. The names of the case study company and its client were not allowed to be disclosed and for this reason, we will refer to the consultancy firm as Company A and to the financial institute as Company B. The two projects were derived from an initial project where Company A used Scrum for the first time in an offshore context, with team members being located in the Netherlands, and in an offshore office in India.

In both projects two different independent applications are built, but there are
3.4 Research methodology

The research was conducted based on a qualitative data analysis approach. Qualitative research refers mainly to the investigation and analysis of personal experiences and behaviors, as well as organizational functions and social interactions [174]. In order to gather the required data for the analysis, we chose to perform semi-structured interviews. In semi-structured interviews, questions can be open-ended allowing a conversational manner, while at the same time, an interview protocol can still be followed [189]. We also had several focus groups with contact people from the company, and we attended their stand-up meetings, demo meetings and closely followed the participating teams.

We interviewed 13 team members from both projects in a period of one month through semi-structured interview sessions. Each session took about one hour. The questions were predefined but open ended. During the period of interviewing the prime investigator worked in the same room with the onshore Scrum team members and also participated in some typical Scrum practices like the daily Scrum meetings and the demo meetings. Through this we could also observe and experience their way of working during the case study. Table 3.1 provides an overview of the different interviewees with their role, current project, site and company.

In the following paragraphs, we describe the case study in detail, by focusing on how Company A works using Scrum in a global software environment, what kind of activities and processes are performed and how they are organized among the
remote teams that participate in the project. We do that using the multi-site governance model, introduced in Chapter 2.

### 3.4.1 Business strategy

In this case study, there are three parties involved; the client (Company B), the onshore provider in the Netherlands (Company A), and the offshore party in India. According to the multi-site governance model, the multi-site business strategy should consider the contractual and legal relationships between the remote sites. In our case study, Company A has a captive center in India, which means that the two remote offices belong to the same firm. A captive center is a subsidiary that a company creates in a offshore location, either by acquiring an existing company or by building one [155].

There is also a legal barrier between the onshore and offshore location; according to the European Union rules and regulations, a person from outside the European Union is not allowed to access financial systems of a company within the EU. This has several implications on the multi-site activities; integration testing can not be performed by people in India and the two remote locations work in different technical environments which they have to keep synchronized on a daily basis.

Additionally, the business strategy between Company A and Company B involves a flexible contractual relationship where change requests can be accepted without extra costs from the client part. This relationship is in line with the Scrum
methodology which “dictates” shorter iterations and more changes can be con-
sidered during development. Finally, the client site is actively participating in
the development process, by having members from Company B within the Scrum
team.

3.4.2 Team structure and composition

In software development, team structure and composition is a critical factor for
good performance [188]. Team size, role descriptions and role distribution are
among those characteristics in distributed teams that can influence team coor-
dination and communication and therefore team performance [121]. In this case
study, we observe several aspects that characterize the way remote and co-located
teams are organized and structured. More particularly, for each one of the two
projects under investigation, there is one, unified Scrum team, with members
from all three engaged parties. According to [86], this type of team structure is
defined as a fully integrated Scrum team, where the team is cross-functional and
the members are distributed across locations. Figure 3.1 illustrates the members
that comprise the Scrum team.

Figure 3.1: A Fully Integrated Scrum Team

<table>
<thead>
<tr>
<th>Onshore Location</th>
<th>Offshore Location</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SCRUM Team</td>
</tr>
<tr>
<td>Company A -</td>
<td>Company A -</td>
</tr>
<tr>
<td>The Netherlands</td>
<td>The Netherlands</td>
</tr>
<tr>
<td>Company B -</td>
<td>Company A -</td>
</tr>
<tr>
<td>The Netherlands</td>
<td>India</td>
</tr>
</tbody>
</table>

The Scrum team is comprised of different roles and responsibilities. In line with
the standard Scrum methodology, there is a Scrum Master whose role is coaching
the Scrum team and guiding it through the Scrum practices and rules. The
CHAPTER 3. FROM RUP TO SCRUM: A CASE STUDY

Scrum Master is a person from Company A in the onshore site. Other members from this site include architects, system analysts and test coordinators. From the offshore side in India, there are mainly developers, testers and a team leader as part of the Scrum team. Additionally, based on the Scrum framework, there is also the role of the Product Owner, the person responsible for managing the requirements list (the product backlog). Here, the Product Owner is represented by a member from the client site (Company B). Other members from Company B include business analysts and requirements managers.

3.4.3 Task allocation

Development is mainly done in the offshore site in India, where the developers and testers are positioned. Architecture and requirements management activities are the responsibility of the Scrum team members located in the Netherlands. Nevertheless, developers and testers from the offshore site actively participate in the requirements decisions, e.g. which requirements will be included in the upcoming Sprint, how to prioritize those requirements etc.

Within the Scrum team tasks and responsibilities are divided into two sub-groups. The first sub-group is responsible for implementing all changes and requirements in the product backlog, and make sure that everything goes according to plan. Part of this sub-group are the developers, the testers and the architects from the onshore and offshore site of Company A, and the system analyst and requirements manager from Company B. The second sub-group of the Scrum team is responsible for preparing the product backlog, deciding what changes are to be implemented and make sure that everything is arranged and agreed upon, before the next Sprint begins. Members of this sub-group include the Product Owner and the business analyst from the client site, as well as a system analyst from Company A. Figure 3.2 illustrates the two sub-groups of the Scrum team, based on their preparation and implementation tasks.

Furthermore, according to [86], Scrum practices need to be extended or modified in order to support global software development teams. In this case study, Company A has extended the standard Scrum framework, to fit the company’s needs. More particularly, they use an additional phase at the end of the Scrum life cycle, during which all activities are summarized, documentation is completed and all the responsible team members ensure that the backlog is completed and the product is ready to be delivered.

Finally, cross-functionality is highly promoted within the teams. Since Scrum prescribes that every team member should be cross-functional this is also applied
3.5 The transition from RUP to Scrum

As mentioned previously, the team members of the current Scrum projects were also participating in older projects where RUP was applied. Therefore, they have experience working in a software global project with both a more traditional methodology (RUP) as well as with an agile methodology (Scrum). In this section, we present the changes that were observed in the transition from RUP to Scrum and the implications that such a transition brought to the projects’ performance. In other words, we compare the current multi-site governance structure of the projects, which is based on the Scrum methodology, with the previous multi-site governance structure based on the Rational Unified Process. Overall, we observe that Scrum improved or eliminated the challenges that the distributed teams were facing when they were working with the more traditional development methodology. One of the onshore interviewees said:
“I am still working on this project because we now use Scrum which works a way better and is a lot more fun to do. If we would have continued working in the traditional way I would have already gone to another project.”

3.5.1 Change 1: Requirements engineering & Customer involvement

One of the major changes of the transition from the RUP to Scrum concerns the process of requirements engineering. During the traditional way of working, requirements were discussed in the onshore site, between people from Company A and the participants from Company B. After deciding on which requirements and which changes were to be implemented they were “throwing the requirements over the fence”, that is they were communicating them to the team in India. As a consequence, the developers and the testers in the offshore site were not participating in the process of requirements elicitation and prioritization. With the current Scrum approach this aspect has changed. All team members, including the developers and testers from India, participate in requirements management, and they can all share ideas and solutions.

The most prominent consequence of such a change is that the team members in India can now gain more domain knowledge. They have the chance to develop personal skills and not only “act as robots”. Additionally, the team feeling has increased, as the people in India feel more close to the their colleagues in the onshore site, as well as to the client site, and their motivation has increased. A developer from the team in the offshore site told us:

“In the RUP model, we worked more isolated and I often felt like I was a robot since I was just developing what I got through documentation, without knowing what it was meant for. Today, we are interacting with the client and I can now also increase my knowledge on their domain which makes the work more interesting.”

Furthermore, while following the RUP, there were three separate teams participating in the project; the onshore site of the consultancy firm, their offshore captive center in India, and the client site. As a result, requirements were first discussed between the onshore people of Company A and Company B, and then the members of Company A were communicating the requirements to the team in India. In other words, the people in the onshore site were acting as the intermediate between the client and the developers. That also meant that whenever the development team in India had questions about the implementation of certain requirements or changes, they first had to ask their onshore colleagues, a process
that can cause more delays and misunderstandings. An onshore interviewee from the client site said:

“In the past a lot of parties were involved in communicating things to India. This adds a lot of noise and delay in the communication process. Now, we can directly talk to India team members and explain them certain things face-to-face. This really improves progress of the project and the quality of the delivered work.”

In the current governance, with the use of Scrum, there is only one, unified Scrum team with members from all three parties involved in the development. That means that the client site is also actively participating in the development process, and they are closely connected with the offshore site. Consequently, requirements are better communicated and understood by all Scrum members. People in India understand better what needs to be implemented and together with their colleagues from the onshore site, they can prioritize the requirements more efficiently. In addition, the members from India can directly communicate with the client site, and as a result communication lines are shorter and less misunderstandings occur. According to an offshore developer:

“Because of agile, we now directly communicate with the client. This helps us understand better what they want and this results in a better product. It is also more interesting to work with the business people (Company B) and hear what they find about the product.”

### 3.5.2 Change 2: Communication

Another change that is observed in the transition from RUP methodology to Scrum practices, is the frequency in the meetings held between the team members and consequently the changes in the frequency of communication. Following the Scrum project life-cycle, the team members meet daily in the stand-up meetings. They have adjusted the time frame of those meetings, from 15 minutes that is prescribed in the standard Scrum practice, to 25 minutes, in order to accommodate the proximity between the Scrum members and the involved delays. An onshore interviewee from the client site said:

“It is easier to handle business when people are working together. I always said that people should work more together and see each other on a daily basis. Now, we are finally doing this through Scrum practices.”

With the daily meetings and the more frequent feedback that the team members can get and receive, misunderstandings between them have been eliminated and
issues are now resolved faster. According to an onshore interviewee:

“Today misunderstandings and impediments are identified much earlier than in RUP. Now we can work on something wrong for maximum one day. In the past, we could be working on a particular task for weeks before any misunderstandings or impediments were identified.”

Additionally, it was observed that through the daily interactions of the Scrum team, the members can synchronize better their activities between the onshore and the offshore locations. It becomes easier to keep track of each others’ progress and status, since the frequent communication allows them to get a daily insight on which part of the project their colleagues work on.

Finally, another change with regard to the communication is the increase of the face-to-face interactions, through their video-conferencing tool. Company A uses a commercial tool called the Eye-Catcher for video-conferences. The EyeâARCatcher is located in separate rooms where team members can talk to each other. In RUP, only certain people were communicating with each other using video-conference. The rest were using emails and instant messaging tools for their communication needs. Now with the daily meetings, where all Scrum members participate, they all participate in the video-conference and they all can see each other face-to-face. This increases trust between the remote and local colleagues and as a result people feel more motivated to collaborate with each other.

“During the Daily Scrum meeting we can see each other through the video-conference tool, which makes it more easy to detect if someone from the offshore site is holding information back. During the meeting it is also easy to break the ice through little jokes for instance. As a result, the offshore colleagues are more comfortable to speak.”

The client site, which in the past had no or very little direct face-to-face contact with the offshore site, now they also appreciate more their offshore colleagues and they show more respect to them since they are participating in the same Scrum practices and see each other daily. An onshore interviewee of the consultancy company said about this:

“In the past there was no face-to-face contact with India. It often happened that someone from the offshore site had some questions about a particular subject and asked this through email to a team member from the client site. As a result, the team member from the client site came to me saying: ‘That person from India with the strange name is asking me strange questions again...’. This reflects how the
3.5. The transition from RUP to Scrum

client site was seeing the India site. Today this has totally changed since India and the client are fully involved and see each other on a daily basis.”

3.5.3 Change 3: Pricing model and internal costs

A change that came with the introduction of Scrum practices concerns the pricing model between Company A and the client (Company B). Because of the agile way of working and the Scrum project life-cycle, there are smaller and shorter iterations between deliveries. Consequently, the members of Company A are more flexible in accepting and incorporating changes that members from Company B request. This arrangement benefits the client site because they don’t need to pay extra for every change they request or for overwork. According to an interviewee from the client site:

“Since we have different agreements with the consultancy company I experience that the consultancy company colleagues are more open for changes from the client. In the past we had to negotiate about the price for every additional feature which was not part of the initial agreement.”

Transition from RUP to Scrum also benefited the internal costs of Company A for communication purposes. In RUP, they were experiencing a lot of communication impediments, which resulted in more internal expenses in order to facilitate communication and collaboration between the remote offices. As a result, the internal costs of Company A were more than the benefits gained from outsourcing to a lower cost country. With the adoption of the Scrum methodology, and the improvement in communication between remote team members, Company A experiences less internal costs and now they can fully profit from the outsourcing benefits.

3.5.4 Change 4: Cross-functionality

Another improvement that was introduced with the change from RUP to Scrum is the cross-functionality of the team members. Team members can be involved in more than one task and the moment they finish working on one task, they can immediately start working on another.

“During RUP it was quite common that the India site had nothing to do since they had to wait for tasks from the Netherlands, but it also happened that they worked the whole weekend to get a certain task
done. Since the introduction of Scrum the offshore site can easily pick up other tasks from the Scrum board. Also, no overworking hours are any longer reported.”

With the traditional way of working, the offshore members did not have the “freedom” to get involved in more tasks, and they had to wait for instructions from their colleagues at the onshore site. This process was causing delays. Now with Scrum, people from the India site can always work on something and therefore their feeling of responsibility and motivation is improved. An onshore interviewee said about this:

3.5.5 Change 5: Delivery time and development pace

During the analysis of the case study, we have observed certain changes that were introduced with the migration from RUP to Scrum which had a negative impact for the projects’ performance. The first change relates to the delivery time. In RUP, delivery was every 6 months while now with Scrum delivery is every 2 months. Because things go faster with the Scrum way of working, there is not enough time for proper documentation. This has a negative effect later in the development phase, because the team members “miss the tracing lines”. In other words, if something goes wrong people find it hard to go back and trace the issue because decisions are not well documented.

Finally, another disadvantage that was introduced with the transition from RUP to Scrum relates to the development pace. Because processes and activities are running now faster than before, the client site finds it hard to follow the pace with which changes are implemented and deliveries are ready. This is because within Company B there are certain departments involved in various aspects of the product such as risk management or quality assurance. Before the software can be released, or before a new feature is introduced, the Scrum team needs approval from these departments. According to their SLA’s this process can take up to 10 days for instance, but during Scrum, people need approval the same day in order to implement requirements and changes instantly and create more business value. Consequently the client site creates delays and bottlenecks.

3.6 Conclusion

The aim of this study was to investigate the transition from a traditional development methodology, to an agile way of working in global software development. We used the multi-site governance model to classify the different global activities
based on three aspects; the business strategy, the team structure and the task allocation. Overall, we observed five different changes in the governance structure and the resulted implications. Figure 3.3 presents the results of our case study.

**Figure 3.3: The transition from RUP to SCRUM**

<table>
<thead>
<tr>
<th>Change 1: Customer Involvement &amp; Requirements Engineering</th>
<th>RUP (Before)</th>
<th>SCRUM (After)</th>
<th>Multi-site Governance Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>The offshore site is involved in the development and testing phases only. They were receiving the requirements and they had to implement them.</td>
<td>The offshore team is involved from the beginning in requirements management.</td>
<td></td>
<td>Task Allocation</td>
</tr>
<tr>
<td>• They have the chance to develop personal skills and not only &quot;act as robot&quot;.</td>
<td>• Team feeling has increased.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The onshore team was acting as an intermediate between the offshore team and the client site, for requirements communication.</td>
<td>There is one unified Scrum team with members from all three engaged parties. There is a direct contact between the client and the offshore team.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Offshore team understands better the requirements</td>
<td>• Shorter lines of communication, less delays.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change 2: Communication</td>
<td>Feedback was not so often. Someone might have been working on a wrong task for weeks or months before they realize.</td>
<td>With the daily meetings they can get daily feedback:</td>
<td></td>
</tr>
<tr>
<td>Only certain people were communicating with each other using the video-conference tool. All other members we using only emails and/or the chat tool.</td>
<td>• Less misunderstandings. Worst case is that someone is working on a wrong task for a day. Issues are identified immediately.</td>
<td>• They can synchronize the work between the remote sites better.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• They can keep track of each others progress and get an insight on which part of the project their remote colleagues are working on.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change 3: Pricing Model &amp; Internal Costs</td>
<td>There was a fixed-price contract between Company A and Company B:</td>
<td>In Scrum there are small and short iterations:</td>
<td></td>
</tr>
<tr>
<td>• It was harder to accept changes</td>
<td>• Company A is more flexible in accepting changes.</td>
<td>• Company B benefits, because they don't need to pay overwork (more cost efficient).</td>
<td>Business Strategy</td>
</tr>
<tr>
<td>• Company B had to pay extra for changes.</td>
<td>• They can synchronize the work between the remote sites better.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company A was experiencing a lot of communication impediments which resulted in more internal expenses in order to facilitate communication and knowledge transfer. Therefore the costs were more than the gains from outsourcing in a lower cost country.</td>
<td>• They can keep track of each others progress and get an insight on which part of the project their remote colleagues are working on.</td>
<td>With the adoption of Scrum, and the improvement in communication, Company A experience less internal costs, which is the initial driver of outsourcing, and that means that now they can &quot;fully profit from the outsourcing benefits&quot;.</td>
<td></td>
</tr>
<tr>
<td>Change 4: Cross-Functionality</td>
<td>In RUP, the offshore site should wait until the onshore team assigns them a new task to do. This caused delays.</td>
<td>The Scrum team is cross-functional. Once the offshore members finish their current task, they can immediately start working in another.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The feel of responsibility grows, motivation is improved and consequently performance is improved. (The members in India are more motivated to ask more questions, than before.)</td>
<td>• The client part (Company B) find it difficult to &quot;keep up with the pace&quot;, especially because for them Risk Management and security are important issues. As a result communication bottlenecks occur.</td>
<td></td>
</tr>
<tr>
<td>Change 5: Development pace &amp; Delivery time</td>
<td>With RUP delivery was every 6 months.</td>
<td>With Scrum now they deliver every 2 months. Things &quot;go faster&quot;, processes and development are running much faster than before.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Disadvantage: The client part (Company B) find it difficult to &quot;keep up with the pace&quot;, especially because for them Risk Management and security are important issues. As a result communication bottlenecks occur.</td>
<td>• Disadvantage: Less documentation is produced: they miss &quot;trace lines&quot; which means that if something is missing or something goes wrong, it is hard to trace it back because it is not documented.</td>
<td></td>
</tr>
</tbody>
</table>

All changes, except one, had a positive impact on the project performance. Scrum methodology introduced a fully integrated Scrum team, with members from all engaged parties instead of separate teams which shortened communication lines among the members and improved requirements communication. Additionally, the daily standup meetings improved performance, because the team members
are now able to identify issues sooner and resolve misunderstandings. The cross-functionality of the teams creates a feel of responsibility and improves motivation of the offshore colleagues. Furthermore, cost benefits were observed as a result of the more flexible way of working introduced by Scrum with shorter and smaller iterations. Finally, there seems to be a negative impact from this transition due to the faster development life cycle of Scrum methodology which also implies limited documentation. Despite the latter negative effect, we can conclude that overall, agile practices can have a positive impact on GSD projects and can help to mitigate many of the well-known challenges of GSD.
Collaboration Structures in GSD

In the past few years, software engineering researchers have adopted social network analysis techniques to understand collaboration patterns in global software teams. In this paper, we investigate current research in global software development where social network theory is used as an analysis technique. We do so through a systematic literature review, where we collect and analyze previous work that adopt a social network perspective in distributed software development. We use the 3C collaboration model to classify our results based on the communication, coordination and cooperation aspects of global networks. Our results reveal two main coordination structures used in distributed teams, namely the clustering and the core-periphery structure. The analysis of the cooperation activities of the global networks reveal differences between planning and practice. Finally, several tools have been identified that aim to improve communication patterns among distributed team members.

4.1 Introduction

In the past few years, software engineering researchers started to adopt social network analysis (SNA) techniques to understand collaboration patterns in global software teams [125]. According to [53], SNA can be used to reveal collaboration patterns that go beyond the organizational charts and the project planning. Additionally, a social network perspective can help locate expertise, raise communities of practice, promote knowledge sharing, and improve strategic decision making across distributed teams [60]. In previous research, social network analysis has been used to help organizations reveal hidden issues among the global software teams [125].
In this paper, we investigate current research in global software development (GSD) where social network theory is used as an analysis technique. We do so through a systematic literature review study where we collect and analyze previous work that adopts a social network perspective in distributed software development. We explore the new insights that both academics and practitioners can gain, by investigating the collaboration of distributed teams in GSD with the use of social network analysis techniques.

The results reveal new aspects in the coordination, cooperation and communication patterns of the remote team members and the impact of these structures to the team awareness and collaboration. The “default” assumption is that remote teams by and large operate independently. The work is coordinated per site; team members communicate a lot with other team members at the same site, and fairly little with team members at another site. The coordination between sites is delegated to special intermediaries, variously known as boundary spanners or brokers. Social network analysis research in global software development reveals interesting new patterns: the core-periphery structure and the appearance of emergent team members. In a core-periphery structure, people in the core – irrespective of site boundaries – interact frequently with each other, and much less so with team members at the periphery. Emergent team members can be people that are not officially part of the team, but yet heavily participate in the team communication, or people that take on the role of boundary spanner or broker unofficially. In both cases, social network analysis reveals the actual collaboration, as opposed to the one from the organization charts.

4.2 Social network analysis in global software development

Social network analysis provides a systematic way of analyzing informal networks by mapping and assessing the relationships between people, teams or organizations [39]. According to [180], an important attribute of SNA is that the relationships between the actors of a network are viewed as interdependent rather than as autonomous units. From a knowledge management perspective, in social network analysis the relationships between the actors are the channels through which communication occurs and information “flows” [180]. For example, previous research suggests that people connected with strong ties, through which a broad set of knowledge flows, are more likely to trigger creative ideas [171]. Cross et. al [39] apply social network analysis to identify points of knowledge creation and sharing within an organization that hold strategic relevance. Taking a social network perspective to understand how a network of people creates
and shares knowledge, makes these interactions within the network visible and as a consequence, related actions and decisions can be taken.

The distributed nature of software development and the use of computer-mediated channels for communication, are particularly attractive for the use of social network analysis to investigate software development activities [53]. An important challenge among distributed teams in global software environments is the aspect of awareness, or in other words, to know who knows what, who works in which part of the project and who is who in the remote teams [121]. In global software development, awareness is defined as a means by which team members become familiar with the work of their colleagues, therefore enabling better coordination between the distributed teams [44].

Figure 4.1: The 3C collaboration model (adopted from [98])

In the field of Computer-supported cooperative work (CSCW), awareness is defined as the intermediate between coordination, cooperation and communication (the 3C Collaboration model; see Figure 4.1) [98]. During communication people negotiate and make decisions, while coordination is the management of these people and their activities. Cooperation is the joint operation in order to execute tasks. Additionally, awareness is the element that offers feedback to the team members about their actions and gives them information about the other participants [70].

In this paper, we use the 3C model to examine how distributed collaboration
can be analyzed from a social network perspective and what are the implications in team awareness. We do so by reviewing current social network approaches which focus on the coordination, cooperation and communication structures of the distributed software teams. We argue that a social network perspective can raise new insights into the distributed collaboration and foster team awareness of global software projects.

### 4.3 Research approach

In order to investigate the potential benefits of a social network perspective in GSD, we performed a systematic literature review. According to Kitchenham & Charters [97], a systematic review is a way to identify, evaluate and interpret previous research on a particular topic, with specific research questions. In this study, we investigate the current approaches to analyze collaboration in global software development, from a social network perspective.

To frame the research questions, we used the PICO criteria, suggested by Petticrew & Roberts [151]. According to these criteria, a research question should include a Population, an Intervention, a Comparison, and an Outcome. In this research, a Population is defined as the area of global software development. The Intervention is the use of social network analysis. The Comparison is the evaluation of the social network analysis as an approach to tackle challenges in global software development. And finally, the Outcome is the analysis of the current practices of social network analysis in global software development and the identification of the new insights we can gain in the distributed collaboration, by using a social network perspective. Therefore, the scope of our systematic review focuses on the following research questions:

1. How are SNA approaches used to analyze collaboration in global software development?
2. What new insights does social network analysis bring to GSD collaboration?

### 4.3.1 Research strategy

According to [97], when starting the search of primary studies, a strategy should be used by defining the keywords, the search query and the digital libraries used as a search source. We have therefore defined a review protocol, based on the aforementioned research questions. The review protocol that we used includes two
4.3. RESEARCH APPROACH

main keywords, namely: **global software development** and **social network analysis**. Based on these search terms, we constructed a list of related keywords, as shown in Table 4.1.

<table>
<thead>
<tr>
<th>Main Terms</th>
<th>Related Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Software Development</td>
<td>Global Software Engineering</td>
</tr>
<tr>
<td></td>
<td>Distributed Software Development</td>
</tr>
<tr>
<td></td>
<td>Distributed Teams</td>
</tr>
<tr>
<td>Social Network Analysis</td>
<td>Social Networks</td>
</tr>
</tbody>
</table>

Based on the derived keywords, a search string was constructed. We used the basic boolean operator **OR** to broaden the search and retrieve papers containing *any* of the terms it separates, and the boolean operator **AND** to narrow the search and retrieve papers containing *all* of the terms it separates.

(social networks OR social network analysis) AND (global software development OR global software engineering OR distributed software development OR distributed teams)

Finally, we applied the search string to the following libraries, as we believe that they provide a confident source for retrieving results in the particular field:

- IEEE Xplore Digital Library
- ACM Digital Library
- Elsevier ScienceDirect
- SpringerLink
- EBSCO
- Wiley Interscience

4.3.2 **Study selection**

As part of the systematic literature review (SLR) research, a study selection was performed in order to filter the results and to make sure that we include only studies that contain useful information for answering the research questions. For this purpose, we have decided upon some inclusion and exclusion criteria that will limit biased or irrelevant results. Table 4.2 shows all the search criteria used during the study selection.
Overall, we are looking for full text papers published in English. We accept papers that describe primary studies, and consequently we reject papers that are a secondary source, that is studies that cite, comment on, or build upon other researches. Additionally, a main inclusion criteria refers to papers that they focus primarily to global software development challenges. Therefore, we include papers that use case studies of software development projects in global/distributed environments.

Table 4.2: Inclusion and Exclusion criteria

<table>
<thead>
<tr>
<th>Inclusion criteria</th>
<th>Exclusion criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Papers that are available online as full texts.</td>
<td>Papers that are not available online as full texts.</td>
</tr>
<tr>
<td>Papers that are written in English language.</td>
<td>Papers written in any language other than English.</td>
</tr>
<tr>
<td>Papers that describe primary studies.</td>
<td>Papers that are secondary sources, i.e. studies that cite, comment on, or build upon other studies.</td>
</tr>
<tr>
<td>Studies that refer primarily to global software development. We include papers that use case studies of software development projects in global environments or otherwise in a distributed manner. We include case studies that refer to development teams that are remotely located.</td>
<td>Studies that do not refer to global software development cases or distributed software teams.</td>
</tr>
<tr>
<td>Studies that use a case study from open source software development, in order to address general global software development issues.</td>
<td>Studies that use a case study from open source projects, in order to address issues in OSS communities only, without taking into consideration the general topic of global software development.</td>
</tr>
<tr>
<td>Studies that use social network analysis metrics and techniques or studies that they take a qualitative approach to social networks perspective.</td>
<td>Studies that they do not use social network analysis metrics and techniques, but other approaches such as basic statistical analysis metrics and techniques.</td>
</tr>
<tr>
<td></td>
<td>Studies that refer to the use of social media applications (such as Facebook, Twitter and others).</td>
</tr>
</tbody>
</table>
4.4. OVERVIEW OF THE RESULTS

The evaluation of the final papers to be selected is based first on the relevance of the title of the paper. If the title does not include any of the selected keywords, or it does not clearly reflect the context of the paper, we look at the abstracts which usually present a brief summary of the paper. If no satisfied decisions can be made for the inclusion or exclusion of the publication, based on the abstract then we scan the full text of the paper (Figure 4.2). These steps were done independently by two researchers. After the filtering by abstracts and by full text, the results were compared and the final papers for inclusion were selected.

Figure 4.2: Search Strategy

4.4 Overview of the results

By applying the search strategy described in the previous section we concluded in 23 primary studies, in total. Given that we are only looking for primary studies in global software development, and the dedicated attention in IEEE Conferences on global software development (ICSE, ICGSE), it is not a big surprise that there was a 56% acceptance rate from the IEEE results. The overall acceptance rate is calculated at 4% (23 accepted papers out of the 606 total results). The total accepted papers are limited in number due to the particular area in which this systematic literature review study focuses and because of the definite research questions that bound our inclusion criteria.
CHAPTER 4. COLLABORATION STRUCTURES IN GSD

Table 4.3: Results of primary papers per online source

<table>
<thead>
<tr>
<th>Source</th>
<th>Total Query Results</th>
<th>Total Included Papers</th>
<th>Acceptance Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE</td>
<td>16</td>
<td>9</td>
<td>56%</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>123</td>
<td>2</td>
<td>2%</td>
</tr>
<tr>
<td>ACM</td>
<td>149</td>
<td>9</td>
<td>6%</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>121</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>EBSCO</td>
<td>94</td>
<td>1</td>
<td>1%</td>
</tr>
<tr>
<td>Wiley</td>
<td>130</td>
<td>2</td>
<td>2%</td>
</tr>
<tr>
<td>Total</td>
<td>606</td>
<td>23</td>
<td>4%</td>
</tr>
</tbody>
</table>

While applying the search query and filtering the results, we identified the following categories of research papers:

1. Papers that address GSD challenges using a social network perspective and a case study from a commercial software company.
2. Papers that address GSD challenges using a social network perspective and a case study from open source software company.
3. Papers that use a social network perspective to address challenges in Open Source Software (OSS) community.
4. Papers that use a social network perspective to address challenges in virtual or distributed teams as a general business issue, and they use a commercial or open source GSD case study.

In the third category, we identified a number of studies that use a social network perspective to address challenges in OSS community. These case studies focus on the design and development of open source software, rather than on the collaboration of distributed teams. For this reason, we choose to exclude this type of papers from our analysis and include only those that explicitly center their attention to global software teams and their collaboration, regardless of the nature of the case study project (open source or proprietary). We admit that, without an explicit link to global software development, many of the issues we identify also surface in publications on OSS.

Additionally, we look at the data collection methods that the selected papers use for their case studies. One of the most commonly used practices is to collect data from surveys (48% of the selected papers use this approach). The researchers collect social network data by conducting interviews with the participants or distributing questionnaires. According to [180], this is the most common method for collecting social network data (i.e. data about the relationships or ties between people within a network), where the questionnaire usually contains questions
about the respondents’ ties to the other people of the network. The second most common data collection method that we observed in our review is from mining various repositories (35% of the selected papers use this approach). In this case, data are collected either from communication repositories such as emails and chat logs, or from software repositories such as change logs and defects libraries. During the analysis, we recognized an essential difference between the two data collection methods; studies that collect data from mining repositories limit the social network participants to the roles of software developers, testers and occasionally architects and/or designers. The reason is that only team members that appear in software documents can be traced as network participants, when data are collected from mining repositories. On the other hand, studies that collect social network data through questionnaires are more broad in the role participation, since more managerial roles of the software development project can be taken into account in the networks analysis, such as project managers, team leaders and others. Finally, we found a small number of papers that used other approaches such as experiments, or a combination of two or more methods to collect data, e.g. questionnaires, documents and observations.

4.5 Analysis of the results

As discussed in section 4.2, a popular model to analyze the collaboration between distributed teams is the 3C collaboration model, introduced in the CSCW field. In this research, we use this model to classify whether the selected papers from the systematic literature review, use SNA to tackle challenges in GSD related to communication, coordination and/or cooperation of the remote teams. We choose the 3C model to classify the results because this theory fits the global software development environments where distributed teams need to remotely collaborate through communication, cooperation and coordination activities, and where team awareness remains a great challenge.

Table 4.4 presents the classification of the 23 selected primary studies, based on which aspect of the collaboration model they focus on. In the following paragraphs, we analyze this selection and classification of the primary studies.

4.5.1 Coordination

In global software development, coordination among the distributed teams is a challenge due to time, geographical and cultural differences [4]. The difficulties of knowing who to contact about what and communicating effectively across sites,
leads to a number of coordination problems [81]. From the analysis, we observe that a social network perspective can bring new insights into the coordination issues in GSD. More specifically, we see that previous researches use social network analysis to investigate the way people are structured within the team and the role that they play in the coordination of the remote groups. Summarizing the results, we identified two main coordination patterns; the core-periphery pattern and the clusters.

**Core-Periphery:** In a core-periphery structure, “a particular group of developers are at the center of the coordination activities. The rest of the developers (in the periphery) seem to rely solely on interactions with the centrally positioned developers, for coordinating their tasks.” [29]. Core-periphery coordination indicates that there is a group of active and interconnected members, independent of geographic location [137]. Figure 4.3a illustrates an example of the core-periphery structure. This coordination structure appears to be in contradiction with the current theory of global software development where remote locations often work as independent groups and there is less cohesion between the teams, due to proximity. Furthermore, the core structure indicates a dense network which in turn, suggests increased collaboration ties and better awareness. A core-periphery structure may also be related to better performance because such structures hold the potential to improve the speed and flexibility with which information diffuses.
4.5. ANALYSIS OF THE RESULTS

within a group. Finally, the identification and analysis of such coordination structures can help manage the dependencies between the work of core members and that of their peripheral colleagues, and vice versa.

Clusters: In this coordination pattern, the network is divided in sub-groups (clusters) which are often based on the geographic location of the teams. Figure illustrates an example of a networks of clusters. A coordination structure of clusters appears to be more in line with the current global software development theories, indicating that members in co-located teams are more connected with each other than they are with their remote colleagues. This coordination structure has a negative effect on the team awareness, since members of a team don’t know their remote colleagues, they don’t know who works on which part of the project and they often don’t know where to find important information, when it is needed. In order to overcome the difficulties that a cluster coordination brings, literature suggests the positioning of a “bridging” role between the remote teams. This intermediate role is the focus of attention of several studies, described in various terms such as “boundary spanners”, “brokers”, “liaisons”, “gatekeepers” and others. Bridging is perceived as a good coordination strategy in the management of distributed collaboration, facilitating team awareness and knowledge management. Bridging roles are also important within a network because they cover the structural holes that might exist between people within the network, or between sub-groups. Global software collaboration is, by its nature, a situation where structural holes may emerge between groups that are geographically, temporally and cultural distant. Chang & Ehrlich conclude that “individuals who are more central can exert more influence by virtue of being connected with other powerful individuals in the network, and have access to more resources than less central counterparts.”

Another role positioning that gains attention in the analysis of distributed collaboration, both in the core-periphery and the cluster coordination pattern, is the “star” which is defined as “the individual who stands in the center of attention”. According to previous studies, a star plays an essential role in the team coordination because he/she is in the center of a network of distributed teams, coordinating activities across the various geographical locations and facilitating knowledge transfer. It was also observed that the role an individual plays in the network can highly benefit his or her performance, mainly because that person can have “ample access to information, rather than controlling the flow of information.” Researching the role positioning of the people that participate in a distributed software development project can help us understand who are those individuals that are more likely to coordinate the distributed activities, how do they perform and what can be done to foster their role within the network.

Coordination is analyzed here from a network perspective. Based on this per-
spective, we can examine the coordination aspect based on the network structure of the teams, i.e. the way people are organized within and across teams and the various roles they can play within the network. Additionally, new insight into the coordination patterns in global software development provide a means for comparison between the different collaboration modes. In our study, we find that the core-periphery is a relatively new structure. Current literature suggests that a core-periphery as a coordination structure has a positive effect in the collaboration between distributed teams. Finally, the clustering structure is a more common structure that conforms to the known disadvantages and challenges of distributed collaboration. Nevertheless, with the right strategic positioning of the team members and the existence of “bridges” or “stars”, these challenges can be surpassed [127].

### 4.5.2 Cooperation

Previous studies examine how social factors, such as organizational culture and history of the relationship between the remote sites, play an important role in awareness of changes in requirements and code that require effective cooperation across sites [56]. Additionally, in another study different approaches to support awareness between teams’ cooperation have been identified, by examining improvements brought to the “shared space” between team members or how they interact using shared artifacts [173]. From a social network perspective, we examine how the analysis of the distributed collaboration networks can bring new insights into the cooperation aspects of the distributed teams.
A main observation in the teams’ cooperation, from a social networking perspective, is the appearance of “emergent” team members. According to literature, emergent members can be of two types; first, they can be people who were not initially planned to participate in the development project but in practice they emerge as contributing members. Second is when a person emerges as the “bridge” between remote teams, even though that position was planned for someone else. A reason for this phenomenon is e.g. the role of the specific person, or his/her position within the team, which might constitute him/her as a better intermediate for the cooperation among the distributed teams. Milewski et. al. note that “most of the bridges we have found in our research and in the literature are not ones that were developed intentionally as part of organization strategy, but units that have grown out of organizational need, or of the team positioning themselves in this role.” As an example in a study conducted among several distributed networks of people working on requirements engineering, results showed that on average, about one third of the team members were emergent roles. In another study, it was observed that although teams were designed to communicate “freely”, i.e. all members from the remote teams could communicate with all of their colleagues in the central locations, in practice only few members emerged as the intermediates between teams at different locations.

Another observation is how cooperation between the distributed teams is facilitated through familiarity and trust. People are more likely to cooperate with each other if they have previously worked together in another project. Furthermore, familiarity can also refer to the awareness of each others’ cultural differences. Previous research in team trust in GSD suggests that when people are familiar with each others’ cultural differences, trust factors between them are higher and consequently cooperation is facilitated; “Developing positive shared experiences across geographic sites leads to stronger team dynamics for distributed software development teams, and stronger team dynamics build trust.” Trust has also been researched as an influential factor on the member’s cooperation performance. Previous work revealed that when individual has a low trustworthiness (indicated by low trust centrality), even if that person is communicating a lot, he/she is not cooperating efficiently with his/her colleagues, as opposed to individuals who communicate a lot but at the same time, they also have high trustworthiness.

The implication on team awareness that these cooperation strategies have, varies. In some cases, it was observed that the emergent members appearing in the projects, despite the initial planning and documentation, negatively impact team awareness. People within the team do not know who they work with, with whom they need to communicate and collaborate and consequently they are not aware of who knows what in the project. On the other hand, case studies were reported,
where despite the appearance of those emergent members, team awareness was high and team members recognized the emergent members and efficiently worked with them. It was also observed that people who are central in the cooperation between the remote teams tend to perceive that the teams are cooperating effectively, which in turn creates a positive environment for cooperation among the distributed team members [31].

4.5.3 Communication

According to the 3C Collaboration model, communication is related to the exchange of messages and information among people [98]. In the field of software engineering, recent studies have turned their focus on the “fit” of the communication patterns between the members of software development teams (distributed or co-located) and the dependencies between the modules/artifacts that each team develops [104]. This “fit” is also referred to as the socio-technical congruence i.e. the match between the social and technical dependencies in a software development project [30]. In global software development, the socio-technical congruence is often studied as a means to analyze and improve the communication and awareness between the remote members, and therefore to improve their collaboration.

During the SLR analysis, we identified studies that use social network analysis to examine the communication patterns of team members, and subsequently analyze and improve the socio-technical congruence of the distributed software development projects. Consequently, we recognized several tools that use SNA techniques as a means to facilitate communication between distributed team members, ease the expertise identification, and increase team awareness. The following paragraphs describe those tools:

**Ariadne**: This tool identifies automatically dependent pieces of code and their authors, and it creates a social network of developers. In other words, it can visualize and present which developer depends on the code of another developer. “The goal of the tool is to automatically identify situations where there is a “mismatch” between the dependency and the communication networks.” [68]. From the experiments conducted using Ariadne, it was noticed that this tool can promote social network awareness among virtual teams and consequently increase familiarity among team members; “Ariadne provides developers an enhanced awareness of interdependencies using a visual approach. Interdependencies are one example of collaborative traces; they describe relationships between developers based on the source code they implement.” [177]. Additionally, it was observed that Ariadne can support managers in monitoring interactions among developers and taking
the necessary decisions.

**Travis**: Travis is a tool that creates traceability networks by linking artifacts, activities and users. Travis also offers different views of the communication network in role-based criteria, i.e. based on whether the user is a developer, a designer, a project manager etc. This option can help eliminate the number of notifications that team members receive, creating an overload of sent messages and avoid over-communication among team members; “The goal here is to reduce the number of notifications that software developers receive because a common problem was the overwhelming flood of notification messages initiated by other software developers or software tools due to changes in the artifacts.” [51]. This has proven particularly useful for the people acting as bridges between teams, or otherwise acting as information brokers. Travis can help these information brokers to manage the large amounts of information they receive from the communication between the collaborating teams; “TraVis provides increased awareness within offshore software development projects based on a broad range of traceability and rationale visualizations that are created with information extracted from the collaborative development platform.” [51].

**Tesseract**: It is a socio-technical dependency browser specifically constructed to show “the social as well as the technical relationships among the different project entities, such as developers, source code, bugs etc” [165]. Tesseract also calculates the communication behavior of the teams, which is the social network of developers determined by the communication records and it monitors progress over time. Additionally, the tool “can highlight the “mismatches” between technical dependencies and the communication patterns of the developers” [165]. This approach can help developers become aware of the other project activities and improve communication with their colleagues; “Some interviewees suggested that the developer-to-developer linkages could serve as a means of creating an awareness of which developers work closely- information that is missing in their distributed work settings.” [165].

**Augur**: Augur “draws views of the network of contributors to a project, relating them according to patterns in their development activity” [50]. Additionally, this network view property can use different visualizations to indicate different features of the relationship between individuals. Similarly to the above mentioned tools, Augur provides a way to examine the communication patterns of the project participants, through the analysis of the technical dependencies of the project; “It provides the technical means to explore the extend to which software artifacts have inscribed into them patterns of interaction and participation.” [50].

Communication relationships have proven important for knowledge sharing and team awareness [164]. An essential feature of the socio-technical congruence tools
4.6 Conclusion

In this paper, we examined how social network analysis can provide researchers and practitioners with new insights into the current global software development challenges. Two different coordination structures of distributed teams in a GSD context have been identified in this SLR study: the core-periphery and the clusters. In terms of team awareness, core-periphery appears the more beneficial structure, yet it does not show up very often in GSD projects. Clusters come with certain disadvantages due to the sub-grouping and the independent work of the teams based on their geographic location. Through the current SNA approaches, researchers have identified strategies that can help tackle these disadvantages, such as the positioning of bridges between the clusters.

Analyzing the cooperation structures from a social network perspective we distinguished two new insights in the current GSD practices: the emergent members and the role of familiarity and trust in distributed cooperation. We observed that through the use of social network analysis new cooperation patterns were identified in projects that were initially planned otherwise.

A social network perspective can help us reveal differences between what is planned and documented within a project, and what happens in practice. These differences can be studied over time, and the findings related to other metrics, such as the number of bugs found, the size of backlogs, and the like, may lead to corresponding interventions.

Finally, communication patterns have been studied through the lenses of socio-technical congruence, in an effort to analyze communication and collaboration between project participants and software artifacts dependencies. We identified several tools that have been introduced in the current literature, with the aim of improving team communication and at the same time to support team awareness and expertise identification.

We conclude that more research is needed in the field of global software development by taking a social network perspective. We argue that this kind of approach may bring new insights into the distributed teams collaboration and promote new solutions and collaboration processes or techniques, to tackle the current global
challenges. Social network analysis can help researchers and practitioners examine the coordination, cooperation and communication structures of the remote and co-located teams, identify potential weak points and support the organization and the decision-making processes of the distributed activities.
Collaboration Structures in Software Product Lines

In this chapter, we present a case study where the SPL organization changed from component-focused teams to product-focused teams. This goes against common advice, which favors a component-based SPL organization. We use social network data to analyze the collaboration networks of the SPL organization, before and after the change. We find that a component-focused organization of the teams leads to a centralized collaboration structure, while a product-focused organization results in a more de-centralized collaboration structure. Perceived advantages of this organizational transition involve a clearer focus and priorities for the teams, as well as less “ping-pong” communication. Some concerns for the future effectiveness of the new SPL organization include the effort to maintain a common platform across products, and the reduced specialization and expertise of team members.

5.1 Introduction

In software product lines (SPL), the organizational structure is an important aspect for the effective collaboration of the teams. Bosch [19] argues that the main reason is because the product line approach creates dependencies between the software components and the teams responsible for those components. Furthermore, problems may occur when the allocation of people to tasks is inadequate, or when decisions are delayed because the responsibilities are not clear and people cannot come to an agreement [153].

The most common architecture of a software product line consists of a software
platform shared by a set of products. Each product configures and uses components of the platform for its own purposes and product specific functionality. From the organizational point of view, there are teams responsible for the products as well as for the platform, and frequently the platform consists of several teams, each responsible for the development of a component.

With the emergence of software globalization, organizational structures become more complex and the collaboration and communication barriers between distributed teams increase. Geographically dispersed teams that develop components within a software product line have significantly more difficulties in implementing the necessary coordination. Furthermore, the distributed nature of software development is particularly attractive for the use of social network analysis (SNA) to investigate software development activities and collaboration structures. According to, SNA can be used to reveal collaboration patterns that go beyond the organizational charts and the project planning.

In this chapter, we present a case study where the SPL organization changed from component-focused teams to product-focused teams. We use social network data to draw the collaboration networks of the SPL organization, before and after the change, and we report on people’s perception on the advantages of the new organization as well as their concerns for its future effectiveness.

5.2 The case study

The study was conducted at Océ, an international company in printing systems. Océ is headquartered in the Netherlands with offices in several European countries, in Asia and Canada. For our case study, we focus on the company’s software product line for wide-format printers. Specifically, we examine the software development activities distributed between teams in the Netherlands (site NL) and a remote office, which we will call site A. From an organizational perspective, site NL and site A are part of the same research and development department of Océ. In a previous work, we have shown how different business strategies between remote locations affect the collaboration of the distributed teams in different ways.

The software platform includes the development of the embedded software, and the printer controller. In our research, we do not consider the development of the embedded software because it is part of a different development department and it is fully developed in site NL.

We focus on the development of the printer controller, which is composed of several components such as controlling print jobs, controlling devices (e.g. scanners),
5.3. RESEARCH METHODOLOGY

and others. By November 2011, which is the beginning of our research timeline, the development of the controller components is distributed between teams in site NL and site A, i.e. a component is either the responsibility of a team in the Netherlands, or a team at site A. Each team includes developers, testers, architects as well as a team leader. At the product level, there are people in site NL, such as product architects, product testers and integrators and a product leader. Figure 5.1 represents the organizational structure of the printer controller development. Finally, in site A there is a development team responsible for desktop applications, and a team responsible for the maintenance activities of the older versions of the products.

Figure 5.1: Organizational Structure up to the end of 2011

5.3 Research methodology

In order to collect the social network data, we used an online survey, which is a common method in social network analysis [136, 180]. We distributed the survey to all project members and we invited the respondents to identify those
with whom they mainly collaborate, in a “free recall” format as described by [180]. Moreover, we performed 10 semi-structured interviews with members from both site NL and site A. For that purpose, we designed an interview protocol to guide the discussions [189], which included two main parts. The first part of the interview was focused on the organizational changes over the past two years. The second part aimed at collecting “opinions” of the interviewees with regard to the aforementioned changes and their ‘perception’ on the effect that these changes brought. We also had several focus groups with contact people from the company, during which we discussed details of our research, presented our findings and interpreted the results.

Finally, the data collection is realized in two different points in time. We started in November-December 2011, where we distributed the online survey for the first time. Two years later, in November-December 2013, we repeated the online survey and at the same time, we conducted the interviews.

5.4 Organizational changes

Over the period of the two years (2011-2013), we identified three major changes in the organization of the product line (Figure 5.2).

Figure 5.2: Organizational Changes Timeline

Starting in January 2012, all controller components are transferred to site A. In September 2013, the teams in site A are re-organized from component-focused to product-focused. Instead of having separate teams developing the controller components, now each team is responsible for a product, within the product line, and the team members work on all components within that product. A few months later, in November 2013, it was decided to also transfer the product integration activities from site NL to site A. Now, teams in site A develop and deliver to site NL a complete, integrated product package. Figure 5.3 presents the new organizational structure.
5.5. ANALYZING THE COLLABORATION NETWORKS

We use the social network data collected from the online survey to analyze the collaboration networks in 2011 and 2013. Overall, the network of 2011 involves 85 people, 39 from site NL and 46 from site A. In 2013 the network consists of 82 people in total, 23 from site NL and 59 from site A. As we expected, the number of people working on the product line from site NL has dropped as most of the development activities were transferred to site A, while the number of people from that site has increased.

Next, we plot the social networks depicted in Figure 5.4. We apply a color-based graphical convention to help with the interpretation; every color refers to a team. The graphical representations (referred to as sociographs from now on) were drawn using the spring embedded layout algorithm. Based on this algorithm, nodes of the network that have small path lengths between them, attract each other and hence they are closer in the graph [72]. The spring embedded layout allows us to get a fair overview of the network and the location of the nodes, by representing which nodes are closer and which nodes are further apart from each other.

Looking at the sociographs and the position of the members within the networks, we observe that people working in the same team are closer to each other. This is apparent as more nodes with the same color are adjacent than nodes with different color. To further support our visual observations, we calculate the number of ties (links) between the teams and across the teams. Table 5.1 and Table 5.2 present...
the results for the networks of 2011 and 2013 respectively, where we see that the number of ties within teams are higher than the number of ties across different teams.

As we explained before, in 2011 the teams were organized based on the different components of the printer controller. By the end of 2013, the teams were organized based on the different products developed within the product line. To better illustrate the different teams, we draw red circles emphasizing the most well represented teams in the network. The results are depicted in Figure 5.5.
Table 5.1: Number of ties within and across teams (2011)

<table>
<thead>
<tr>
<th></th>
<th>CC1</th>
<th>CC2</th>
<th>CC3</th>
<th>M</th>
<th>A</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC1</td>
<td>38</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>CC2</td>
<td>6</td>
<td>46</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>6</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>CC3</td>
<td>5</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>28</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>46</td>
<td>5</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>P1</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>11</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

CC= Controller Component Teams, M= Maintenance Team, A= Applications Team, P= Product Teams

Table 5.2: Number of ties within and across teams (2013)

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>A</th>
<th>M</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>44</td>
<td>18</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>18</td>
<td>64</td>
<td>12</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>7</td>
<td>12</td>
<td>20</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>18</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

P= Product Teams, M= Maintenance Team, A= Applications Team

In the sociograph of 2011, we can see several controller component teams, while in the sociograph of 2013 the teams represent now the different products in the product line. Moreover, the organizational changes have not affected the Applications team and the Maintenance team, which are represented in the sociographs of both 2011 and 2013.

An interesting observation here is that the network of 2011 exhibits a centralized organizational structure. Particularly, we see that the different component teams are in the periphery of the network, while the members of the product teams (product leaders, product testers and integrators) are positioned in the center of the network. This is a fair representation of the way of working in 2011, as all component teams were delivering to the product teams for integration and testing.

On the other hand, we observe that the network of 2013 exhibits a more de-
centralized organization. We do not see a central team anymore, and the collaboration lines in the network run between the product teams, as well as the Applications and Maintenance teams. This organizational structure reflects the initial intentions of the company to re-organize the teams in order to become more independent in terms of decision-making and consequently more effective. Although, we do not have enough data to support whether the new, decentralized organization meets the company’s expectations, previous research suggests that decentralization provides team members more freedom to make decisions and pay closer attention to each other [162].
5.5. ANALYZING THE COLLABORATION NETWORKS

5.5.1 Advantages of the new organizational structure

As we described in Section 8.4, we also collected data from the interviews, where people expressed their opinions with regard to the organizational changes implemented in the product line. Based on this, we have identified two perceived advantages.

Clear priorities and better team alignment

Previously, the members of each controller component team were fully or partly assigned to work on a product depending the priorities. As a result, the members of other products will “complain” that they do not get enough “attention” from the component teams. With the new organization, the focus of the teams is on the product and the team priorities are based on the product’s deliveries. An exemplary quote is:

“Each team member works for a product now, so that they can focus on the product; whether it is a color printer, or a black & white printer, whether it is a high volume, or low volume. That gives a focus to the team members, a better feeling when we have a deadline.” (Product architect, site NL)

Alignment and coordination between the teams has also improved:

“Sometimes there were misunderstandings and one team would develop something, another team would develop something else but we needed three months to discover that we were not aligned. Now everyone is sharing the same meetings, so people working on different components are talking every day. At least alignment is easier and faster than before, and we expect some benefits in this area.” (Team Leader, site A)

Less “ping-pong” communication

The second advantage regards the communication between team members, as well as between locations. With the old organizational structure, the component teams delivered to the product teams for product integration and testing. Consequently, there was a lot of “ping-pong” communication about defects in the product and it was not always clear which component team was responsible. With the new situation, integration is realized within the same team.
“Personally, I find that this worked very well, as it got rid off a lot of back and forth emails about unclarity and communication. For example, before if I had a question on a feature and I would say: ‘hey it’s not working’ then I might get an answer like: ‘In my part it’s ok, you have to go the other component team’. And of course these are all different people and some are more responsive than others. And now I only have one contact person and that really helps.” (Product leader, site NL)

5.5.2 Concerns for the future effectiveness of the new organizational structure

In addition to the perceived advantages, we also identified two important concerns for the future effectiveness of the new organization.

Maintain the platform consistency

An important aspect that was commonly accepted among interviewees as a concern for the future of the product line is to maintain a common platform between the products. Working for the same issues in different product teams, implementing changes that are not consistent with all products are among the concerns that team members should regard, as a result of the new organizational structure.

“The trap is that maybe we loose the platform approach, meaning that at some points maybe we will have different versions for each product of the same software component. So that’s something we have to address because we really want to keep the platform approach. So what we develop for one product can be easily used by another one too.” (Team Leader, site A)

Loss of specialization and expertise

Finally, another expressed concern was the potential decline of specialization of the team members. The new organizational structure allows people to work on all components of the controller hindering the development of expertise knowledge. As a team leader from site A told us:

“The idea is that you have a team capable of doing everything, so you do not have specialized people. So you have a pool of developers and then you assign them what is more urgent. The difficulty then
is that you need to be a specialist to do, for example, user interface work, or to work on a pdf or postscript interpreter, and it takes years to learn. So the idea as it is now is that you do not have specialized teams. It takes time to learn, and now we are still searching on how to work with that.” (Team Leader, site A)

Taking a closer look at the current literature in SPLs, these concerns are referenced among the main advantages of migrating from a product centric to a product line organization. The high quality achieved through the reuse of a shared core and the improved productivity due to specialization of the teams are considered important aspects in software product lines [17].

5.6 Related work and discussion

Although Bosch [15] stresses the importance of organizational structures in software product lines, to date little in-depth research has been done in the field. In 2001 Bosch [15] described four organizational structures namely, the development department, the business units, the domain engineering and the hierarchical domain engineering. These organizational structures have also been adopted by [141].

Furthermore, the authors in [153] propose several matrix-oriented structures of a software product line. Finally, in a later work several approaches have been introduced on how to organize an SPL, taking also into consideration the geographic dispersion of the teams [18].

All the aforementioned approaches have one common denominator; the division of work and the creation of the teams within the SPL based on the component distribution of the development activities. In this paper, we present a different approach where the case study company changed their component-focused teams to product-focused teams. The underlying organization was transformed from a centralized structure to a more decentralized one.

Currently, we find that there is not enough information at hand to further analyze such a transition. This is partly, because our research was conducted in a period where the organizational changes in the case study company were still recent and as a result the situation was in a state of flux. And it is partly, because of the limited research data to compare with, caused by a lack of in-depth research in this area.

Nevertheless, the interviews we performed with the people working in the product line acknowledge two major advantages of the new, decentralized organization;
the clearer priorities and better alignment of the teams, as well as the limitation of the “ping-pong” communication. Finally, two main concerns emerged, which are the effort to maintain a common platform across products and the potential loss of specialization.

5.7 Conclusion

In this paper, we use social network data to study an SPL organization. We do so at two different points in time. In between, the organization changed from a component-focused organization to a product-based organization. The component-based organization exhibits a centralized collaboration character, with a central team at the product line level. The product-focused organization shows no such centralized collaboration structure.

Due to the limited research in the area of organizational structures in software product lines, the conclusions we draw from our results are limited. Yet, our work can trigger further research in the field of SPL organization. Using social network analysis techniques future case studies may supplement as well as extend our current observations.
Introducing Transactive Memory Systems

In this chapter, we present the theory of transitive memory systems. Additionally, we propose a twofold approach for measuring TMS in global software development; first, a structural approach where we use SNA techniques to analyze the structure of TMS. Second, a process approach where we measure the different processes (transactions) performed in a GSD environment, through a latent variable model.

6.1 Introduction

A transactive memory system is a shared cognitive system for encoding, storing, and retrieving knowledge between members of a group. Within TMS, the group members have a collective awareness of each others’ specialized knowledge domain which helps them identify and locate where expertise resides within the group [83]. Another characterization of a TMS is as a shared understanding of “who knows what” within the group. Transactive memory systems have been mainly studied at the group level and the organizational level [159]. A multi-level approach, which also includes the individual level analysis of TMS was proposed by Yuan et. al in [196]. Particularly, Yuan et. all [193] argue that transactive memory systems emerge from the actions performed by the individuals and that individual-level outcomes are significantly related to team-level TMS [194, 191, 192].

Studies in global software development turned their attention to the importance of TMS in the collaboration and communication of the distributed teams [89]. One of the main concerns in GSD that steered the attention towards TMS is the lack of knowledge that people have about the expertise domain of their remote colleagues. Faraj & Sproull [65] proposed a model to test expertise coordination and their empirical results showed that factors such as locating and sharing
expertise knowledge significantly improves team performance. In another study [32], the authors identified a positive relationship between different dimensions of TMS (such as knowledge differentiation, knowledge location, knowledge credibility) and the communication quality, as well as the performance of the project members. Coordination of geographically dispersed teams is also influenced by how well people are aware of each others’ expertise [64].

According to Wegner et al. [183], transactive memory has two components: an organized store of knowledge (TM structure) and a set of transactions/processes (TM processes) related to knowledge management processes of encoding, storing, and retrieving. The TM structure is a knowledge representation of members’ shared understanding of who knows what. TM processes are the mechanisms by which the group coordinates members’ learning and retrieval of knowledge, so that the knowledge can be applied to group tasks.

In this chapter, we examine both the structure of transactive memory systems and the processes involved. We also adopt a multi-level analysis of TMS, where we use results from the individual and dyadic actions to examine the performance of TMS at the group-level. In the following sections, we present our approach in measuring TMS.

### 6.2 TM structure and social networks

In order to identify and examine the structure of transactive memory systems, we adopt a social network perspective. Social network theory has proven beneficial in analyzing transactive memory systems. For instance, Yuan et al. [194] study the social networks of 18 organizational teams and proposed that within a knowledge sharing network, the strength of ties between the members of the network can influence expertise awareness. Their findings suggest that although people in contemporary organizations can learn about each other’s areas of expertise through, for instance, direct interpersonal communications or expertise directories, it is through communication ties that employees can gain actual access to diverse expertise. Another study [195] shows that in distributed teams, homophily as a social characteristic is an essential factor for driving the formation of network ties. Additionally, Borgatti et al. [12] developed a model in which the probability of someone in a network seeking information from another person is a function of (1) knowing what that person knows, (2) valuing what that person knows, (3) timely access to that person and (4) the perceived cost associated with accessing that person.

In chapter 4 we discussed two prominent collaboration structures that emerge in
GSD; clusters and core-peripheries. Additionally, the results from the systematic literature review showed that the position of the members within a network plays an important role in their collaboration. Centrality measures are popular SNA measures for identifying members’ position within a network [69]. In this chapter, we propose the use of the following three SNA approaches to analyze transactive memory structures:

1. Clustering, which looks into whether actors of a network tend to form subgroups (clusters) and how strong those clusters are.

2. Core-periphery structure identifies those actors that are densely connected and form the core of the network, as well as those actors that are loosely connected with the rest of the network and are part of the periphery.

3. Centrality measures seek to identify team members that occupy central positions and therefore, they have an influential role within the structure.

6.3 TM processes as a latent variable model

There have been several proposals on how to measure transactive memory processes [3, 115, 190]. Lewis & Herndon [117] suggest that transactive memory can be either assessed directly or indirectly as a latent variable. Direct measures allow us to draw conclusions about TM processes as a whole, while indirect measures are useful in settings where TM processes cannot be clearly assessed. The choice on how to measure TM processes lies mainly on the case study at hand [117].

In this thesis, we are interested in examining the effects that different global software governance decisions have on transactive memory systems. In the field of global software development, transactive memory processes are not clearly defined and hence, we adopt an indirect measure of TMS.

Furthermore, according to Lewis [115] a measure of TMS must meet two basic criteria. The first is to be theoretically consistent with Wegner’s conceptualization of TMS that reflects the cooperative processes and transactions of memory used in a group [132]. The second criterion is to align the measurements with the settings of the field we are interested in studying. Following these suggestions, we consider previous studies and conceptualizations of the theory around transactive memory systems and we create a latent variable model to measure TMS in global software development. We choose to create a TMS model, tailored to the concepts of GSD, rather than use an existing one such as the model proposed by Lewis [115], which provides general manifestations of transactive memory systems. The rest of this section describes how we operationalized TMS theory and
CHAPTER 6. INTRODUCING TRANSACTIVE MEMORY SYSTEMS

how we used well-established concepts to create a latent variable model for TMS processes.

6.3.1 TM operationalization

Table 6.1 presents the operationalization model that we used in order to express transactive memory as a latent variable. As suggested by Lewis [115], the main drivers of our proposed model are the three basic processes of a transactive memory system: directory updating, information allocation and retrieval coordination [182]. Encoding the information in a memory system implies that group members keep their memory directories updated with regard to what others in the group are likely to know, or otherwise what is the others’ expertise domain. Storing the information in a transactive memory system entails the process of allocating knowledge to the person whose expertise will facilitate its storage. Additionally, in a knowledge network information is retrieved based on knowledge of the relative expertise of the individuals.

Table 6.1: TMS Operationalization

<table>
<thead>
<tr>
<th>Expertise Awareness</th>
<th>Directory Updating</th>
<th>Information Allocation</th>
<th>Retrieval Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1. I am aware of this person’s area of expertise.</td>
<td>Q2. I am aware of the knowledge that this person needs for his/her job.</td>
<td>Q4. It is easy for me to access this person.</td>
<td></td>
</tr>
<tr>
<td>Accessibility</td>
<td></td>
<td></td>
<td>Q5. The information I receive from this person is important for my job.</td>
</tr>
<tr>
<td>Credibility</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication Frequency</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Previous studies propose that specialization, coordination and credibility are aspects that can reflect the cognitive processes within a transactive memory system [65, 115, 133]. Transactive memory systems exist when members of a group have differentiated and specialized knowledge domains, they are aware of the expertise of each other (Expertise awareness), and they can rely on each other for this specialized knowledge (Credibility) [3]. Additionally, in order to share that specialized knowledge team members need to develop coordination activities that will allow the knowledge to be shared and transferred easily across the network (Accessibility).

Finally, we incorporate into our proposed TMS model the idea of communication frequency as an important aspect that needs to be assessed if we want to
reflect the existence of TMS within distributed teams \[12\]. Especially in global software development, communication frequency is often regarded as a barrier to the collaboration and knowledge transfer between remote team members \[140\]. In settings where people collaborate across different geographical locations, it becomes easier for a collective memory system to be built and maintained when communication frequency is high.

6.4 Conclusion

In this chapter, we proposed a twofold approach in measuring transactive memory systems in GSD. Figure 6.1 illustrates that approach. First, we examine the structure of the transactive memory systems using social network analysis techniques. Particularly, we propose the analysis of the network structure based on the two popular approaches of core-peripheries and clusters \[13\]. We also propose the use of centrality measures to identify the most important (central) members of the networks.

Figure 6.1: Measuring Transactive Memory Systems

<table>
<thead>
<tr>
<th>Transactive Memory Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM Structure</td>
</tr>
<tr>
<td>Social Network Analysis based on:</td>
</tr>
<tr>
<td>- Clusters</td>
</tr>
<tr>
<td>- Core-peripheries</td>
</tr>
<tr>
<td>- Centrality measures</td>
</tr>
</tbody>
</table>

Second, we examine transactive memory systems from a process point of view. Specifically, we propose a latent variable model where we capture the different processes (transactions) enabled in such systems. We synthesize our operationalization model using existing literature and proposed methods in the field of TMS, and we tailor this approach to the field of global software development.

In the next chapters, we use our proposed method to identify and analyze transactive memory systems in GSD companies.
The Effect of Governance on TMS

In this chapter, we seek to identify how different governance decisions (based on business strategy, team configuration, task allocation) affect the structure of transactive memory systems as well as the processes developed within those systems. We present an analytical method to examine GSD governance decisions and their effect on transactive memory systems. Our method can be used from both practitioners and researchers as a “cause and effect” tool for improving collaboration of global software teams.

7.1 Introduction

Over the years globalization of software development activities turned into a common practice. Factors such as the coordination and synchronization of the activities across locations and different time zones, the communication and the knowledge management between distributed teams became familiar among scholars and practitioners. And while research in global software development (GSD) evolves and new practices emerge [47], Herbsleb [78] notes that “there is little reason to expect that these factors will diminish”.

Our purpose is not to diminish those factors influencing GSD collaboration, but rather identify them and use them as a tool for a “cause and effect” analysis. Particularly, we are interested in investigating how different decisions that companies take on how to govern their GSD activities, affect knowledge management processes and more specifically, the development of transactive memory systems. In the following paragraphs, we elaborate on that purpose.

As discussed in chapter 2 with the continuous and evolving strategies in global software development there is a turn of interest towards the challenges and the
CHAPTER 7. THE EFFECT OF GOVERNANCE ON TMS

key issues of managing GSD activities [93, 55, 176]. As a result, global software development governance turned into an emerging field of research, as a subfield of information technology (IT) governance. The purpose of GSD governance is to identify those aspects that are necessary for an effective coordination and collaboration among distributed teams. As Bannerman suggests [3], “governance is the infrastructure needed to ensure the satisfaction of direct and indirect stakeholders”. For instance, when engaged in global activities one can decide to create a captive center in a remote location, or to make a “client-supplier” contract with an external partner. Managers also need to decide how to structure their teams and how to allocate tasks among geographically dispersed members. What parts of the projects will be outsourced to the remote partners, and which parts will remain onsite? What kind of responsibilities will be delegated to the offshore partners and how much to delegate? These are all questions that frame the governance structure that a company builds for its global activities.

Furthermore, different working practices, geographic proximity and/or legal barriers between remote offices influence the development of transactive memory systems [121]. In the previous chapter (chapter 6), we introduced transactive memory, the kind of memory that the team members develop and which helps them identify “who knows what” within the team [182]. In order for the team members to develop such a memory, they have to engage into various transactions through which expertise knowledge is created, shared and stored. As a result, a cognitive system (transactive memory system) is created where members are aware of each others’ expertise domain and they are able to access it, update it, share it and facilitate its storage.

Coming back to the main purpose of this study, we pursue the following research question: How do GSD governance decisions affect transactive memory systems? We examine four case studies, in two multi-national companies and we identify their governance structure. Based on the different governance decisions that each case study employs, we report on the differences in the development of transactive memory systems. We present those differences as a “cause and effect” analysis in order to explain how different GSD governance modes (a set of governance decisions) affect the collaboration and communication of the distributed teams.

7.2 Project overview

The research was conducted in two multi-national companies. The first company, which we call Eco, is developing printing systems on a global scale. Eco is headquartered in the Netherlands, with offices in more than 100 countries and over 20,000 employees. Research and development departments work on hardware
as well as software innovations, and they are distributed across nine different countries.

The second company, which we call Ricon is active in the telecommunications field in more than 180 countries. Ricon employs over 100,000 people worldwide, offering network platforms, telecom services and multimedia solutions for mobile communications. During our case study, we worked with the offices located in the Netherlands, which are involved in hardware logistics, as well as in research and development of multimedia solutions.

In the next sections, we elaborate on the projects that participated in this research. In total, four GSD projects collaborated with us, two from Eco and two from Ricon. We are setting the scene of each project, and we identify the governance mode (the set of governance decisions) selected by each case.

Table 7.1 summarizes the governance modes that we have identified, based on the business strategy, team structure and composition, and task allocation.

<table>
<thead>
<tr>
<th>Governance mode 1: Eco BU1</th>
<th>Offshore insource. Past experience based on offshore outsource.</th>
<th>Boundary spanners are emergent. Offshore teams are hierarchically structured.</th>
<th>Close Collaboration: architectural decisions and development tasks are shared. System testing and integration remain in-site.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Governance mode 2: Eco BU2</td>
<td>Offshore outsource.</td>
<td>Boundary spanners are emergent. Offshore teams are hierarchically structured.</td>
<td>Strict separation: part of the development is outsourced. The rest of the activities remain in-site.</td>
</tr>
<tr>
<td>Governance mode 3: Ricon PR1</td>
<td>Offshore outsource.</td>
<td>Boundary spanners are designated.</td>
<td>Strict separation: development tasks are entirely outsourced. The rest of the activities remain in-site.</td>
</tr>
<tr>
<td>Governance mode 4: Ricon PR2</td>
<td>Offshore outsource. Past experience with offshore insource.</td>
<td>Boundary spanners are designated.</td>
<td>Close collaboration: development tasks are entirely outsourced. The rest of the activities are shared.</td>
</tr>
</tbody>
</table>

7.2.1 Setting the scene: Eco

Our research focuses on two business units of Eco company (we refer to them as BU1 and BU2), which specialize in high-end printers. The development of the software used in both business units is distributed between the main site (site NL) and two remote sites: site A and site B. Figure 7.1 illustrates the distributed collaboration between the locations involved in BU1 and BU2. The software
includes units such as accepting requests, controlling print jobs, rendering images, controlling devices (e.g. scanners) as well as local and remote user interfaces.

Figure 7.1: Eco’s remote collaboration

Within BU1, site NL and site A work closely together and, from a legal perspective, they are part of the same research and development department of Eco. Following the taxonomy in [169] the defined business strategy is offshore in-source, with near geographic distance and small temporal distance. Consequently, the remote offices are allowed to freely exchange knowledge and the distributed colleagues collaborate with each other without any legal barriers limiting their communication. Additionally, the collaboration between site NL and site A is a long-term collaboration and the two sites have worked together for many years, as they moved from a client-provider relationship to a legal merge.

On the other hand, site B is a different legal entity from site NL and the collaboration and communication between remote offices is rather restricted. According to the taxonomy in [169] the business strategy between site NL and site B can be defined as an offshore outsource strategy where geographic distance is far but temporal distance is small. Members from site NL are allowed to share information with members from site B that refers only to the tasks they are working on together. General information and knowledge about other parts of the system can’t be shared with team members of site B.

In a typical team structure, a team leader heads the project and is responsible for its planning, realization, and successful completion. Teams also comprise architects who transform the high-level specifications into detailed technical features and pass them to software engineers who implement the code. There are also system integrators, responsible for integrating the different software units. The communication between distributed teams is not strictly regulated by designated people. Team members are encouraged to proactively share knowledge and initiate communication with their remote colleagues. Furthermore, in site NL there is a flat organization and structure of the teams, without strict reporting lines.
between the architects, software engineers and testers. Team structure together with organization processes differ at the other two remote locations, site A and site B. The different tasks and responsibilities between the roles is clearly defined, documents and issues must be officially accepted before being processed.

Finally the collaboration between remote sites in BU2 involves the strict separation of tasks and activities among remote (and co-located) members. The team leader of the specific software unit allocates development tasks to software engineers. Several criteria are applied in deciding which tasks should be sent to site B such as previous experience, i.e. whether a software engineer has worked on a certain task before, or the complexity of the task. Finally, only a small part of the project is developed in site B, while the majority of the project remains in site NL. Within BU1, tasks and activities are allocated across teams and across locations in a more interconnected way, where all members work interdependently for the completion of a project. In the beginning of each release cycle, team leaders and architects from all sites create a plan for the next release and discuss the requirements to be implemented. At the end of each release, the work developed and tested from the remote sites is shipped back to the NL site for integration and system and product testing.

### 7.2.2 Setting the scene: Ricon

At Ricon’s site, we examined two projects that we refer to as PR1 and PR2. PR1 focuses on the development of a communication platform, and it is a collaboration project between the Netherlands (site NL) and two development teams from a remote location, site C. The second project, PR2 involves the development of various applications that run on the platform developed by PR1. Furthermore, PR2 project concerns the distributed collaboration between site NL and two development teams in a remote location, site D (Figure 7.2). Both site C and site D are officially external partners, i.e. they are independent software companies with which Ricon has contractual agreements. Following the taxonomy in [169] we define the business strategy with sites C and D as an offshore outsource strategy, where geographic distance is far. Additionally, temporal distance is small for site C, and large for site D.

The decisions for initiating a distributed collaboration with site C and site D differ, and it is important to take them into consideration for the rest of our analysis. Particularly, the collaboration with site C started based on cost saving decisions, where Ricon was looking for a software development provider in a low-cost location. On the other hand, Ricon had a development center (captive center) in site D, which closed down. After that, the company in site D re-
opened as an independent software vendor, employing the majority of the people working there before. The teams of site NL initiated a distributed collaboration with the new company on a client-provider agreement. The collaboration has an important advantage for the teams in site NL as they work with teams from site D that have already a domain-specific knowledge on telecommunications and more specifically on Ricon’s projects.

Both projects follow the SCRUM methodology and therefore they employ roles such as scrum masters and product owners. Apart from scrum masters, a development team also involves software developers, testers, and architects. In line with the SCRUM methodology, Ricon keeps small development teams of five to seven people. Additionally, both projects have dedicated teams of testers responsible for testing activities at the system-level, as well as for system integration. Within Ricon’s projects we also observe the existence of pre-defined teams and members responsible for bridging the communication across locations. Particularly, in PR1 there is a dedicated team in site NL, called team-in-between, which is responsible for the coordination of the activities with site C. Team-in-between includes architects and testers. Similarly, in PR2 there are specific members (architects and product owners) from site NL, who are responsible for the coordination of the activities with site D.

To regulate their collaboration with the external partners, Ricon employs what they call internally “steering strategies”. These steering strategies depend on the strategic importance and/or the complexity of the development work. Managers decide how much of the responsibilities to delegate to the external partners, ranging from having a team executing only software development tasks, to a self-sustaining and self-managing team. During our case study, within PR1, site C responsibilities are limited to development activities, with a scrum master in every team to coordinate their work. All other decisions and tasks (architecture, system testing, integration) remain in site NL. On the other hand, within PR2,
7.3. DATA COLLECTION AND ANALYSIS

site D has more responsibilities including participating in software architecture decisions and requirements engineering processes. At the same time, site NL is also training members from site D for system testing and integration.

7.3 Data collection and analysis

To analyze the projects and answer our main research question, we use both quantitative and qualitative data. Surveys are the most common method for collecting social network data [136, 180]. Additionally, the purpose of our study is to collect data regarding transactive memory processes, which refer to the cognitive memory of individuals. Therefore, we choose to collect social network data using an online survey. We distributed the survey to all project members of Eco and Ricon. The questionnaire invited the respondents to identify those with whom they mainly communicate, in a “free recall” format as described by [180, pg.45]. Additionally, we asked the respondents to value the relationship (on a 5-point Likert scale) with each one of their connections, based on the questions Q1-Q6 presented in Chapter 6. The end result is a set of data that represents who communicates with whom, and a set of six valued relationships for every connection. Overall, BU1 involves 101 people (response rate 52%) and BU2 consists of 70 people (response rate 47%). Additionally, we collected data from 54 and 48 people from Ricon, working on PR1 and PR2 respectively (response rates 50% and 42% respectively).

As we previously discussed, TMS is measured as a latent variable characterizing the link between two people in the network (respondent i - connection j), based on six items, i.e. the six questions of the survey. We performed a reliability analysis, to estimate the degree to which the selected items measure the underlying construct of transactive memory. A proper analysis for assessing the reliability of scales is Cronbach’s alpha coefficient [37]. Cronbach’s alpha measures the internal consistency, which is the extent to which all items in a scale can reliably express a latent variable. Cronbach’s alpha coefficient for our data set is calculated at .708. This is an acceptable value of internal consistency and hence, we can conclude that our model can be reliably used to measure transactive memory as a latent variable.

Collecting social network data allows us to express TMS both as a dyadic relationship between two people, as well as an individual characteristic of each one of the members within a team. In chapter 6 we explained that our indirect way of measuring TMS is guided by the three basic transactions of directory updating (DU), information allocation (IA) and retrieval coordination (RC). Directory updating is constructed using one item (Q1), information allocation is constructed
CHAPTER 7. THE EFFECT OF GOVERNANCE ON TMS

using items Q2, Q3 and finally retrieval coordination is constructed using items Q4, Q5, and Q6. Therefore, we calculate TMS at the dyadic level according to Eq. (7.1).

$$TMS_{dyadic(i,j)} = \frac{Q1 + \mu(Q2 + Q3) + \mu(Q4, Q5, Q6)}{3}$$

Subsequently, we need a TMS score that can be attributed to every person in the network, and which will give us a value for the transactive memory at the individual level. We calculate the individual TMS based on the total scores of $TMS_{dyadic}$ that an individual has with his direct connections, divided by the total amount of people in the network, N. Using N as the denominator in Eq. (7.2), we take into consideration the unequal amount of direct connections that someone has, by weighting higher those with more connections and rating lower those with less connections.

$$TMS_{individual(i)} = \frac{\sum_{j=1}^{j} TMS_{dyadic(i,j)}}{N}$$

Moreover, we had several focus groups with contact people from the company, during which we discussed details of our research, presented our findings and interpreted the results. The data collected from the survey were further supported and triangulated by means of interviews with team members from both companies. We interviewed members from all locations in every project involved. We chose a variety of roles in order to have an accurate representation of the situation in each project. Such roles include software engineers, testers, integrators, architects, and various management roles like team leaders, project managers, scrum masters, product owners etc. In Table 7.2, we present details about the number of interviews and the participants from each project.

<table>
<thead>
<tr>
<th>Table 7.2: Number of interviewees</th>
</tr>
</thead>
<tbody>
<tr>
<td>BU1 (Eco)</td>
</tr>
<tr>
<td>BU2 (Eco)</td>
</tr>
<tr>
<td>PR1 (Ricon)</td>
</tr>
<tr>
<td>PR2 (Ricon)</td>
</tr>
</tbody>
</table>

During the interviews, we used a protocol with open-ended questions to guide the discussions. The protocol had three main areas of discussion: (a) background information about the interviewee, (b) questions about the governance of the
7.4. CLUSTERING

We begin the analysis with plotting the sociographs of all four cases, from both Eco and Ricon (Figures 7.3 and 7.4). We also applied a color-based graphical convention to help with the interpretation of the sociographs; every color refers to a location, with orange denoting people from site NL, blue is site A, black is site B, yellow is site C and finally green refers to site D. For the structural representation and analysis, we used UCINET [14] a software package for the analysis of social network data, and NetDraw [11] for the graphical representations. The graphical representations (referred to as sociographs from now on) were drawn using the spring embedded layout algorithm. Based on this algorithm, nodes of the network that have small path lengths between them, attract each other and hence they are closer in the graph [72]. The spring embedded layout allows us to get a fair overview of the network and the location of the nodes, by representing which nodes are closer and which nodes are further apart from each other.
Examining the sociographs, we first notice that people, from all networks (BU1, BU2, PR1 and PR2), form subgroups (clusters). It is also apparent that clustering is based on the geographic location of the team members. For instance, we see that people from site NL are closer to each other, forming their own cluster, while people from sites B, C and D have their own separate clusters. BU1 presents a slightly different picture, where people from site NL and site A are well interconnected, showing a weaker tendency to a location-driven separation.

Location-driven clustering is a common structure in global software development settings, for example in [31, 27, 62]. However, our analysis brings up a new observation on the connectedness of these clusters. For instance, we notice that in BU2, site B has only few connections with site NL, while in BU1, site B is tightly connected with the rest of the network (Figure 7.3). Similarly, looking at Ricon’s
sociographs, we notice that the clusters in PR2 are better connected, while the clusters in PR1 have fewer links (Figure 7.4). Based on the interviews, we find that business strategy is a common denominator affecting clusters’ connectedness. The remote locations in BU1 were working together in the past, in a client-provider contract, and now they merged into the same legal entity. The remote locations in PR2 were part of the same legal entity, and now they work in a client-provider relationship. Both cases provide the distributed teams with years of experience working together, as well as gaining specific-domain knowledge (on printing systems for Eco, on telecommunication systems for Ricon). Eventually, the business strategy in both cases allow the teams to freely share knowledge and create effective collaboration patterns:

“I have worked with site B for quite some years now, and they are beginning to transform from a third party that executes what we want, into a partner and that makes the system also better.” (System integrator, site NL, BU1)

On the other hand, in BU2 and PR1 people from site NL are engaged into a relationship with external, remote providers from which they “buy” services. As a result, there are legal differences between the remote offices that impede information allocation and retrieval coordination transactions. External service providers do not have direct access to all the knowledge stored in site NL and they rely on the information that their colleagues selectively send them. Few exemplary quotes:

“If there are problems in sharing information, it is due to the legal issues. They couldn’t send us, because they are not allowed to. [...] in theory, we have access to all kind of information that they have access to. I say in theory, because in practice we are different legal entities and there are intellectual property issues, so they cannot disclose everything and this creates annoyances” (Team leader, site B, BU1)

“We can have only access to that code which we are developing. But if you want to go into details about the platform, then we don’t have authorization to access that code or the documentation. If we are facing an issue in that part, then we have to raise a ticket and they will take it to the respective team.” (Scrum Master, site C, PR1)

Additionally, our results suggest that task allocation can also affect clusters’ connectedness. Specifically, we learn that in BU1 and PR2, remote teams closely collaborate with each other, taking architectural decisions together and commonly defining requirements. This can be the result of an offshore insource relationship, arguing that since there are no legal barriers and knowledge sharing constraints, teams are free to closely collaborate with each other. But close collaboration can also occur in a offshore outsource relationship, as we see in BU1
between site NL and site B. Eventually, a close collaboration including shared decision making among distributed teams, and the delegation of responsibilities to the remote sites create frequent interactions between the distributed teams and promote team awareness:

“They are more mature, and we also give them more on architecture. But we work with them, let them make a proposal and then we can look at it, with our architects here, and we decide together with them, like ok this should be the approach we have to take. So they are more involved in the whole process and they really understand what we want from them.” (Product owner, site NL, PR2)

To further support our observations, we examine transactive memory systems from a process-driven approach, using the latent variable model described earlier. In order to compare TM processes between remote colleagues, we use the $TMS_{dyadic}$ scores, and we distinguish between pairs where members work from different locations. Table 7.3 presents the descriptive statistics of all projects. We note that on average co-located colleagues have higher scores than remote colleagues. When we look at these results independently for every project, they do not come as a surprise. Nevertheless, our purpose here is to analyse TMS scores and examine whether that difference is significant, and whether our previous observations on clusters’ connectedness influence that difference. Therefore, we compare the $TMS_{dyadic}$ scores of the remote colleagues that belong to networks with tightly-connected clusters (BU1, PR2), with the scores of the remote colleagues that belong to networks with loosely-connected clusters (BU2, PR1). The results support our expectations, suggesting that TMS processes are better developed between remote colleagues in BU1 and PR2 ($M = 3.29, SD = .90$; $t(198) = 2.50, p < .05$) than between the remote colleagues of BU2 and PR1 ($M = 2.90, SD = .80; p > .05$).

Table 7.3: TMS dyadic - Descriptive statistics

<table>
<thead>
<tr>
<th>Company</th>
<th>Project</th>
<th>Dyadic relationship</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eco</td>
<td>BU1</td>
<td>Remote colleagues</td>
<td>3.62</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Co-located colleagues</td>
<td>3.77</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>BU2</td>
<td>remote colleagues</td>
<td>3.32</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>co-located colleagues</td>
<td>3.72</td>
<td>0.41</td>
</tr>
<tr>
<td>Ricon</td>
<td>PR1</td>
<td>remote colleagues</td>
<td>2.55</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>co-located colleagues</td>
<td>2.83</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>PR2</td>
<td>remote colleagues</td>
<td>2.69</td>
<td>1.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>co-located colleagues</td>
<td>2.90</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Summarizing our results, we observe that an offshore insource collaboration eliminates legal barriers between remote teams and hence, facilitates accessibility and
expertise awareness. Additionally, when distributed locations closely collaborate with each other, e.g. taking architectural decisions together, and commonly defining requirements, communication frequency and credibility are enforced. Such governance decisions have eventually a positive effect in transactive memory systems; location-driven clusters are tightly connected and the members develop better transactive memory processes.

7.5 Centrality and Boundary spanners

The second step in our analysis is to look at the centrality measures of the networks. The centrality measures attempt to describe the position of the actors in the network and identify the most prominent, the most central ones. Here we discuss three of the most important centrality measures: degree centrality, betweenness centrality and closeness centrality.

Degree centrality is the number of direct connections that an actor has. The idea behind degree centrality measure is that an actor who has many direct connections is also well connected and therefore that actor can be a central point within the network. In terms of communication and knowledge transfer, actors with high degree centrality are considered focal points of communication and major channels of information. Therefore, the degree centrality of an actor is an important index of the actor’s potential communication activity and ability [69].

Betweenness centrality measures how likely an actor is to be the most direct route between two other actors in the network. Particularly, betweenness centrality is the extent to which an actor lies between two others, acting as a broker or a bridge, managing and mediating information [8]. Additionally, actors with high betweenness centrality usually seem to have control over the information that flows between others in the network [135].

Closeness centrality is a measure of how fast someone can reach everyone else in the network. The idea behind closeness centrality is that an actor is central if he can quickly reach and interact with all others in the network [180]. Actors with the shortest paths to all others have a good overview, a good insight, of what happens in the network. Another study [69] suggests that actors with high closeness centrality can be very productive in communicating information across the network and when they are engaged in problem solving they can provide efficient solutions.

In order to identify the most central members of the networks from Eco and Ricon, we look at the people that score higher in all centrality measures (we use the top ten scores from degree, closeness and betweenness results). Next, we
annotate those who score higher in all centrality measures for every project. In Figure 7.5 we illustrate the most central members with an enlarged-size node. We also indicate the area within the network where the central people are positioned, by drawing a red circle around them.

Figure 7.5: Boundary spanners

Looking at the centrality results, we observe that the most central members within the network are those that bridge the location-driven clusters. In other words, it appears that the most central people within each network are the members that act as boundary spanners between locations. As it was recognized in a previous study, brokerage involves activities such as building sustainable relationships, resolving conflicts, negotiating and motivating [184]. Inevitably, boundary spanners become central members within their networks.

As we previously described, Eco does not have designated boundary spanners. The results of the centrality scores reveal that the most central people in both
BU1 and BU2 are the team leaders, and consequently the boundary spanners between locations. From the interviews, we learn that due to the hierarchical structure of the offshore teams, the role of team leader emerges as the “interface” between remote locations:

“They are surprised when I’m asking them something without involving their team leader. So they are not used to it. ‘Do I have to answer this? ...my coach hasn’t agreed on that. â€œ should I spend time on this?’ ” (Software Engineer, site NL, BU2)

On the other hand, Ricon has designated teams and members, which formally act as boundary spanners between locations:

“We have the product owner and the software architect as main contacts in site NL. And almost every request or every question should go through them.” (Architect, site D, PR2)

Whether boundary spanners are formal or informal, they require a substantial amount of real-time interactions and a tolerance for frequent interruptions in order to bring remote colleagues together, create new relationships and share knowledge \cite{74,112,96}. Hence, boundary spanners develop a system-generic knowledge that allows them to have a general overview of the status of each project and the tasks each team is working on \cite{8}. Based on these suggestions, we would expect that members who are more central in the network, would also have better transactive memory systems developed. If someone has many direct connections, he is close to many others, and he is in between the communication of many others, then that person is also better aware of the knowledge created, shared and stored across the network.

In order to assess the levels of TMS for these roles, we compare the results of the centrality measures with the scores in $TMS_{\text{individual}}$. The results support our expectations, indicating that there is a positive, significant correlation between centrality measures and TMS processes (Table 7.4). Note that the scores of closeness centrality are reverse. The actual scores measure “farness”, hence people with high scores are far from all others in the network, while people with low scores are close to all others in the network.

<table>
<thead>
<tr>
<th></th>
<th>$TMS_{\text{individual}}$</th>
<th>mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>.763**</td>
<td>6.37</td>
<td>4.45</td>
</tr>
<tr>
<td>Closeness</td>
<td>-.534**</td>
<td>209.97</td>
<td>85.43</td>
</tr>
<tr>
<td>Betweenness</td>
<td>.166**</td>
<td>68.27</td>
<td>95.85</td>
</tr>
</tbody>
</table>

**correlations significant at $p < .001$
CHAPTER 7. THE EFFECT OF GOVERNANCE ON TMS

Summarizing the results, we observe that within the composition and structure of GSD teams, there are boundary spanners who have a better overview of the network’s activities and bring remote teams together by facilitating knowledge sharing. Due to these attributes that boundary spanners have, they become central members within their network and they develop better transactive memory processes.

7.6 Network core

The last step is to look at the core-periphery structures of the networks. The core of the network tends to have a more central position, which indicates that the actors of the core are not only very well connected with each other but they are also well connected with the rest of the actors in the periphery. Core-periphery structure in Open Source Software projects (OSS) is sometimes referred to as the onion structure [129]. The core developers are at the center of the onion, they contribute most of the code and manage the design of the project [111]. At the periphery of the onion structure, there are developers who commit less to the project by delivering e.g. only bug fixes, use-cases or just testing new releases. The core of OSS projects is important because many essential tasks for the software development processes (such as system architecture, conflict resolution, leadership etc.) are performed by team members that belong to that core [12].

<table>
<thead>
<tr>
<th>Site</th>
<th>Site NL</th>
<th>Site A</th>
<th>Site B</th>
<th>Site C</th>
<th>Site D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>BU1</td>
<td>14</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>22</td>
</tr>
<tr>
<td>BU2</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>14</td>
</tr>
<tr>
<td>PR1</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>13</td>
</tr>
<tr>
<td>PR2</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>17</td>
<td>20</td>
</tr>
</tbody>
</table>

Borgatti & Everett [13] were the first to propose methods for detecting core-periphery networks. They developed two models; a discrete model that defines two classes within the network, the core and the periphery. This model distinguishes the network members between those that belong to the core, and those that belong to the periphery. The second model is a continuous model where every member in the network is assigned a measure of “coreness” indicating how close the actor is to the core.

For our analysis we use both the discrete and the continuous model. We use
the discrete model to locate the core members of every project. The results are summarized in Table 7.5. We then use the continuous model to assign all members a coreness value, as an individual attribute characterizing their position in a core-periphery structure.

Interestingly, the analysis of the core reveals a pattern between core members’ distribution across locations and task allocation. Figure 7.6 shows that there is a correspondence between the tasks and responsibilities of the offshore locations, and the proportion of the core members in the network, originating from these locations.

Figure 7.6: Task allocation and core composition
Starting from the first case (BU2) in Figure 7.6, we notice that site B is responsible for only a part of the development tasks and consequently, the entire core of the network consists of members from site NL only. In the second case (PR1), development is the sole responsibility of the offshore location, a decision that gives site B a small share (20%) in the network core. The third case (BU1) shares architectural decisions as well as development tasks with the remote teams, and therefore people from site A occupy almost half (40%) of the network core. The last case (PR2) shows us the “other side of the coin”, where the majority of the core (80%) consists of people from the offshore site as most of the tasks and responsibilities are delegated to the external partner (site D).

As Cataldo & Herbsleb [28] point out “those individuals on the core, not only perform a critical communication role, but also they are the top contributors to the actual development effort. In other words, the individuals at the core are also highly productive”. We expect that core members in GSD collaborative networks, are better aware of the current activities within the system, who works on which part of the project and of the current status of the project. Comparing the coreness scores with the $TMS_{individual}$ scores, we find that there is a positive, significant correlation between them ($r = .588, p < .001$). These results suggest that members closer to the core of the network develop better transactive memory processes.

Summarizing our findings, we observe that deciding how to allocate (delegate and/or share) tasks among distributed teams is an indicator of where expertise resides (in-site or off-site). Core members are well connected with all others in the network, they are actively involved in GSD activities and consequently, they develop better transactive memory processes.

### 7.7 Lessons learned

There are no “one size fits all” solutions in global software development. Every company is different and every person is different. With this in mind, we proposed a way to identify and analyze the different governance decisions of GSD companies, and observe how such decisions affect transactive memory systems. Our findings are summarized in Figure 7.7. We present four different governance modes, based on decisions in business strategy, team structure and composition, and task allocation. Subsequently, every governance mode affects transactive memory systems on two aspects: on TM structure, which is expressed through social network measurements of clustering, centrality and core-periphery, and on TM processes measured as a latent variable.
## The Governance Effect

<table>
<thead>
<tr>
<th>Governance Modes</th>
<th>Business Strategy:</th>
<th>Team Structure &amp; Composition:</th>
<th>Task Allocation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Offshore insource. Remote locations are part of the same legal entity.</td>
<td>Boundary Spanners are emergent. Offshore locations tend to be more hierarchically structured.</td>
<td>Close collaboration. Architectural decisions and development tasks are shared. System testing and integration remain at the main site.</td>
</tr>
<tr>
<td></td>
<td>Offshore outsource. Remote locations are part of different legal entities.</td>
<td>Boundary Spanners are emergent. Offshore locations tend to be more hierarchically structured.</td>
<td>Strict Separation. Only part of the development tasks are outsourced. The rest of the activities remain in-site.</td>
</tr>
<tr>
<td></td>
<td>Offshore outsource. Remote locations are part of different legal entities.</td>
<td>Boundary Spanners are designated. Assigned team members bridge remote colleagues.</td>
<td>Strict Separation. The development tasks are outsourced. The rest of the activities remain in-site.</td>
</tr>
<tr>
<td></td>
<td>Offshore outsource. Remote locations are part of different legal entities.</td>
<td>Boundary Spanners are designated. Assigned team members bridge remote colleagues.</td>
<td>Close collaboration. The development tasks are outsourced. Architectural decisions and System testing and integration are shared.</td>
</tr>
</tbody>
</table>

- **Clustering:**
  - Smaller tendency for geographically-based clusters.
  - Tightly connected clusters.
- **Centrality:**
  - Team leaders are the central people in the network.
- **Coreness:**
  - 80% of the core members are in-site.
- **Within tightly connected, geographically-based clusters, remote colleagues develop better transactive memory processes, than remote colleagues within loosely connected, geographically-based clusters.**
  - The more central a person is in the network, the better developed his transactive memory processes.
  - The closer to the core of the network a person is, the better developed his transactive memory processes.
CHAPTER 7. THE EFFECT OF GOVERNANCE ON TMS

From a practical point of view, we propose an analytical method that companies may benefit from and use it as a “cause and effect” tool. For instance, GSD companies can use it to evaluate their business strategy with the offshore teams and whether it leads to strong or weak ties between remote teams as well as the influence it might have on their knowledge transactions. Likewise, managers can analyze their team structure and composition and decide whether to assign formal or informal boundary spanners and who should become central member of their network. One might also benefit from the core-periphery methods to determine how to allocate tasks, how much to delegate to the offshore sites and what tasks to share with the remote teams.

From a theoretical perspective, we contribute to the body of knowledge in GSD research in several aspects. First, we present a systematic way of analyzing governance structures in GSD and their effect in transactive memory systems. Particularly, we suggest the use of the GSD governance model to analyze multi-site activities based on three aspects; namely here the business strategy, task allocation, and team structure and composition. We also apply a two-fold approach to analyze transactive memory systems; a process and a structural approach. As GSD research evolves and the attention to knowledge management challenges increases, we believe that our contribution will benefit future research.

Second, we point out the importance of cluster connectedness. As we previously discussed, location-driven clustering is a common phenomenon in GSD, but little attention has been paid so far in the significance of the connectedness of such clusters. Through our research, we find that business strategy decisions such as pursuing an offshore insource approach, creates tightly-connected clusters, which in turn leads to better developed transactive memory systems. Third, we contribute to existing research on the role of boundary spanners in GSD, by comparing cases between formal and informal boundary spanners. We find that within the composition and structure of GSD teams, brokers play a central role as they have a better overview of the network’s activities, bring remote teams together and facilitate knowledge sharing. Finally, we present a novel mapping between task allocation and core-periphery structures in GSD networks. We show how from sharing small tasks with the offshore teams to delegating and sharing more responsibilities, affects the composition of the network core.

7.8 Conclusion

In this paper, we examined how different governance decisions influence transactive memory systems. We analyzed governance decisions according to business strategy, task allocation and, team structure and composition. Additionally, we
used a two-fold approach in analysing transactive memory systems, based on the structure as well as the processes involved in such systems. We conclude with the presentation of a “cause and effect” method that we can use to examine how different governance decisions (the cause), affect in different ways the structure and the processes of TMS (the effect).

Our results suggest that looking at the connectedness of the distributed locations (clusters) involved in the GSD activities, we can examine how well (or not so well) remote colleagues are aware of each other’s expertise knowledge. Furthermore, identifying the central members of a GSD collaboration network we can locate those members that span the boundaries across locations and bridge expertise knowledge between distributed teams. Finally, we learned that analyzing the composition of the core of a GSD network, we can explore where the most active members of the project are located.

We contribute to the body of knowledge in the field of global software engineering, by proposing a different analysis of multi-site collaborations. We merge the business and governance aspects of a GSD with the evergrowing challenges in knowledge management and expertise awareness, using as a “glue” social network analysis techniques. Finally, we encourage future research towards the further enhancement and development of the use of transactive memory systems and social networks in global software development studies.
A common collaboration structure in global software development (GSD) is clustering, wherein people tend to be closer to others with whom they share common characteristics. Clusters often create barriers in communication, coordination and expertise awareness between remote teams, restraining the development of transactive memory (TM). In order to overcome such barriers, the role of brokers has emerged. In this paper, we examine the role of brokers as facilitators in the development of transactive memory. We use social network theory to analyze the collaboration of an EU-funded project, where development teams come from different partners and different locations. Our results suggest that task-based clusters emerge and that project members who coordinate activities as well as those who contribute to the code development act as brokers. Our empirical evaluation shows that clustering has a negative effect on TM and that brokers can moderate that effect.

8.1 Introduction

Many studies in global software development (GSD) focus on the collaboration structures of distributed teams. Following Conway’s law [35], for instance, previous studies examine the relationship between the architectural designs of the system, the task allocation and the teams’ collaboration (e.g. [80, 79]). Other researches focus on the analysis of GSD collaboration structures using social network analysis (SNA) techniques. For example, Damian et. al. [44] use social networks to examine the dynamic nature of membership in the different stages of...
a multi-site software development project. Social network analysis has also been applied in the collaboration patterns of requirements engineering among multiple development locations [124].

To date, a common collaboration structure in global software development is clustering [123]. Clusters are formed within a network of interconnected members, where some members are closer to each other, creating dense subgroups. These groups are homogeneous, as their members share certain common characteristics. In global software development, the most prominent characteristic is geographic distance [31], meaning the people from the same location communicate more and collaborate closer with their co-located colleagues than with their remote ones. In other collaborative environments, such as in open source software development communities, a common characteristic that brings people together into clusters can be the project that the members are working on, or shared interests such as language technologies [7].

Clusters may lead to communication and collaboration challenges, limiting expertise awareness ([81, 31]), and consequently restraining the development of transactive memory (TM). Transactive memory is the memory that members of a group develop for encoding, storing and retrieving expertise knowledge within their group [182]. As Faraj & Sproull note [65] “teams must be able to manage their skill and knowledge interdependencies effectively through expertise coordination, which entails knowing where expertise is located, knowing where expertise is needed, and bringing needed expertise to bear.”

In order to overcome such communication and collaboration challenges caused by clusters, literature suggests the positioning of a bridging role between the remote teams [127]. These bridges or brokers bring different groups together, facilitate communication, improve knowledge sharing and often promote innovation [113].

In this paper, we examine the role of brokers as facilitators in the development of transactive memory (TM). We perform a case study on a multi-site project in order to answer the following questions:

1. What kind of clusters emerge in the project’s collaboration structure?

2. Which members play the role of brokers and what are their characteristics?

3. To what extent do brokers facilitate the development of transactive memory?
8.2 Clusters: a collaboration structure

In the previous chapters, we discussed two main collaboration structures; clusters and core-peripheries. In a core-periphery structure, “a particular group of developers is at the centre of the coordination activities. The rest of the developers (in the periphery) rely on interactions with the centrally positioned developers, for coordinating their tasks” [29]. Several studies also suggest that core-periphery structures lead to effective collaborations because they can improve the speed and flexibility with which information diffuses (e.g. in [43, 186]).

Despite the positive effects of core-periphery structures on the communication and knowledge sharing among distributed teams, such structures are more popular in open source software (OSS) communities, than in commercial GSD collaborations [186]. The main reason is that OSS communities are less hierarchically organised and members’ participation is mainly voluntarily. Up to date, the most common collaboration structure in commercial GSD is clustering. Team members that share common characteristics, tend to “attract” each other and create dense sub-groups. In global software development, proximity is the most common characteristic, bringing co-located members closer to each other. For instance, a previous study [122] presents the collaboration structure of two multi-site software development projects where location-based clusters are formed and some of them are tightly connected with many communication ties between the locations, while others are loosely connected with only few communication links.

Another characteristic that drives clustering is task allocation, i.e. people that work together on the same task or project tend to be closer to each other. For example, Chang & Ehrlich [31] examine how members of global software teams create sub-groups based on the project they are working on, and that the communication between those sub-groups is often coordinated by a group leader or an architect. In the same vein, several other studies build up Conway’s law [35], suggesting that the collaboration structures of software teams represent the software modules of the system, eg. [69, 80, 53]. Task-based clusters are also often found in OSS projects, where contributors create subgroups based on the projects they are participating in (eg. in [120, 41, 187]).

Clustering creates several challenges in the communication and sharing of expertise knowledge, and particularly on the development of transactive memory. The tendency to create homogeneous clusters poses a special challenge to knowledge-intensive collaborations as the aim of such collaborations is to produce knowledge and to disseminate it quickly and effectively [114]. For instance, members of location-based clusters do not always know their remote colleagues, and as Rogers & Lea [161] observe “the lack of social presence reduces the pressure to
respond to an information request, so that a request from a remote individual is less likely to get handled quickly”. Similarly, members of project-based clusters are often less aware about the domain expertise of their colleagues in other teams, so that they do not always know the right person to contact, or where to find appropriate information sources [81].

In order to overcome such challenges, literature suggests the positioning of a bridging role between the clusters. This intermediate role is the focus of attention of several studies, described in various terms such as “boundary spanners”, “brokers”, “liaisons”, and “gatekeepers”. The next session elaborates on the role of brokers.

8.3 The role of brokers

Several studies have analysed the characteristics of brokers. Williams [184], for instance, describes the broker as a person who must be capable of building sustainable relationships, negotiating and managing interdependencies. Brokers must also be able to balance conflicting demands of multiple groups (clusters) with differing expectations [152]. When brokers bridge clusters that have conceptually separated boundaries (e.g. geographic proximity, organizational or departmental boundaries, etc.) can also referred to as boundary spanners [119, 72].

An interesting classification of the role of brokers is that brokers can either be formal or informal [96]. Formal brokers are dedicated members that are particularly appointed to play the role of mediator between teams. For example, in [122], communication between remote teams is defined by the hierarchical organization of the offices, and consequently team leaders and project managers act as bridges between locations. Informal brokers are members that emerge as mediators between clusters, due for example to that person’s expertise. Milewski et. al. note that [127] “most of the bridges we have found in our research and in the literature are not ones that were developed intentionally as part of the organisation strategy, but units that have grown out of organisational need, or of the team positioning themselves in this role”. Informal brokers are mainly common in open source projects where communities are created based on flat-hierarchies and self-organization [172] [67].

Brokerage is also perceived as a good coordination strategy in the management of distributed collaboration [8]. Brokerage roles are important within a network because they cover the structural holes that might exist between people, or between groups of people. Global software collaboration is, by its nature, a situation where
8.4 Research methodology

Our research is based on data collected from members participating in an EU-funded research project, under the seventh Framework Programme (FP7)\footnote{http://www.opener-project.org}. The project focuses on the enhancement of natural language processing techniques. It is a two-year project (July 2012-July 2014) and it involves the collaboration of six partners from three locations; Spain, Italy and the Netherlands. Our case study was performed at the end of the first year of the project (June-July 2013).

The work of the project is divided into five main tasks, which are distributed among the partners. Table\ref{tab:tasks} presents the tasks as well as the partners responsible for them. We refer to the two partners from Spain as S1, S2 and to the two partners from the Netherlands as N1, N2. Similarly, we indicate the two partners from Italy as I1, I2.

Our analysis is based on collecting social network data through a web-based survey. The questionnaire invited the respondents to select those with whom they mainly communicate, from a list including all active project members. Additionally, we asked the respondents to value the relationship (on a 5-point Likert scale) with each one of their connections, based on the questions Q1-Q6 presented in Chapter\ref{chap:measures}. The end result is a set of data that represents who communicates with whom, and a set of six valued relationships for every connection. Overall, the project involves 37 members and we had a response rate of 92% (only 3 people did not respond to the survey).
8.4.1 Measuring transactive memory

As we discussed in chapter 6, we measure transactive memory as a latent variable characterizing the link between two people in the network (respondent $i$ - connection $j$), based on six items, i.e. the six questions of the survey. In the previous chapter (chapter 7), we applied the same survey to collect social network data and we have introduced the formulas to calculate the dyadic and individual transactive memory processes. In this chapter, we use the individual scores of transactive memory.

We also performed a reliability analysis, to estimate the degree to which the selected items measure the underlying construct of transactive memory. Cronbach’s alpha coefficient for our data set is calculated at .84. This is an acceptable value of internal consistency and hence, we can conclude that our model can be reliably used to measure transactive memory as a latent variable.

8.4.2 Measuring clustering and brokerage

In order to identify clusters within the network, we used a graphical representation (referred to as sociograph from now on) to illustrate the collaboration structure of the project. We draw the sociograph using the spring embedded
8.5. ANALYSIS & RESULTS

layout algorithm [14]. Based on this algorithm, nodes of the network that have small path lengths between them, attract each other and hence they are closer in the graph [72]. The spring embedded layout allows us to get a fair overview of the network and the location of the nodes, by representing which nodes are closer and which nodes are further apart from each other.

We also calculated the clustering coefficient of every member and of the entire network using the methods proposed by Watts [181]. The clustering coefficient of a single member is calculated based on the density of the member’s local neighbourhood, i.e the extent to which a member’s direct connections are also connected with each other. The clustering coefficient of the entire network is the mean of all member’s clustering coefficient.

Finally, to identify the brokers in the network, we used the measure of betweenness centrality. Betweenness centrality is a commonly used measure in social network analysis to identify central members, and more specifically “the extent to which the agent can play the part of a broker or gatekeeper” [166]. To calculate betweenness we used the method proposed by Newman [135], which counts the times a person lies in between the shortest communication path of two others.

8.5 Analysis & results

In this section, we answer our research questions and present the results of our analysis. First, we examine the coordination structure of the project by identifying the kind of clusters that emerge. Particularly, we are interested in identifying the common characteristics that drive project members closer to each other, creating subgroups. Next, we look at the members of the network that play the role of brokers, bridging the different clusters. We examine the characteristics of the top brokers such as their role within the project and their contribution. Finally, we perform a regression analysis in order to study how brokerage moderates the effect that clustering has on transactive memory.

8.5.1 What kind of clusters emerge in the project’s collaboration structure?

In order to identify the clustering tendency of the network, we begin our analysis with the overall clustering coefficient measure. The clustering coefficient of the network is calculated at 0.57, which indicates a moderate tendency of the project members in subgrouping (clustering coefficient ranges from 1, indicating total clustering to 0, when there are no clusters). Additionally, we plot the sociograph
of the project (Fig. 8.1), using the spring embedded algorithm. To help with the interpretation of the network, we applied a colour-based graphical convention where every colour represents a partner:

- Partners from Spain: S1 red, S2 blue.
- Partners from Italy: I1 yellow, I2 green.
- Partners from the Netherlands: N1 black, N2 orange.

Figure 8.1: Project’s collaboration structure: Overview of clusters

To understand what kind of subgroups emerge, we look at the sociograph in Fig. 8.1. The most prominent clustering tendency is based on the tasks the partners are working on. Particularly, we see that the two partners working together on management and integration tasks (S1, N2) are closer to each other in the network’s structure. Similarly, the two partners responsible for the development of language technologies (S2, N1) also “attract” each other. Finally, the partner responsible for data harvesting (I1) creates a separate cluster within the network’s structure.

An interesting observation is the position of the second partner from Italy (I2). This partner is represented by only two members and they are responsible for the design and system architecture. Architectural decisions affect all other partners working on the different parts of the system and consequently, these two
members are positioned between the integration tasks, data harvesting and the development of language technologies.

Summarising the results, we observe a moderate tendency in task-based clustering. Previous studies suggest that this type of clusters can be found both in commercial GSD projects where Conway’s law is represented, as well as in open source communities. As Prause et. al. note “it is the nature of EU projects that company, culture, and language boundaries run through the midst of a project task. It is not one partner responsible for requirements engineering, but several. Similarly, several partners will do architecture design, implementation work and testing.”. Therefore, analysing the coordination structure of this research-driven, GSD project, we observe that there is a moderate tendency for task-based clustering.

8.5.2 Which members play the role of brokers and what are their characteristics?

Next, we calculate betweenness centrality for all members in the project. In order to identify the most important brokers in the network, we look at the top 10 members with the highest betweenness scores. Fig. 8.2 illustrates who are those members and where they are positioned within the network structure. Our first observation is that the brokers are located towards the middle of the sociograph, spanning the boundaries between the task-based clusters.

Examining the characteristics of the most important brokers, we see that five out of ten are partner leaders (we marked them in the sociograph with the letter L). Partner leaders are appointed as key members representing each partner, aligning and coordinating the activities of their teams with the rest of the project participants. Also, a central member of the network is a person appointed as the technical coordinator of the project (marked in the sociograph with the initials TC). The technical coordinator is responsible for keeping an overview of the components’ development process and the implementation solutions. Another interesting observation is the position of the person appointed as the administrative coordinator (depicted in the sociograph with the letters AC). The administrative coordinator is responsible for the project’s deadlines (i.e. make sure everybody delivers on time), communicating documents to the EU commission as well as leading project meetings.

Additionally, we look at the code repository of the project at GitHub to find which members contribute to the code development. As Fleming & Waguespack point out “technical contribution increases a member’s likelihood of becoming a leader in an open innovation community”. Table 8.2 presents the contributors for
the language technology and the data harvesting components. We notice that two top contributors (ora and bur) to the development of the language components are also among the top brokers of their task-based cluster. Similarly, the top contributor to the development of the components for data harvesting (tes) is also a top broker in his cluster.

Table 8.2: Code Contributors

<table>
<thead>
<tr>
<th></th>
<th>Contributors</th>
<th>Number of commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Technology</td>
<td>ora</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>bur</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>ana</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>iga</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>moc</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>asi</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>civ</td>
<td>1</td>
</tr>
<tr>
<td>Data Harvesting</td>
<td>tes</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>nag</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>raf</td>
<td>5</td>
</tr>
</tbody>
</table>
Brokers can either be formal or informal. Summarising our results, we find that both formal and informal brokers exist within the collaboration structure of this project. Particularly, we observe that formal brokers are those members appointed to coordinate the project activities and they have a more “managerial” role, such as team leaders, technical and administrative coordinators. On the other hand, informal brokers emerge among those members that have a more “technical” role within the project. As Lee & Cole [109] also note, in a community-based project roles emerge in the process of performing tasks, as opposed to firm-based projects where roles are planned and members are assigned to carry out tasks. In the same vein, informal brokers emerge due to the expertise knowledge that certain members develop through their technical contribution.

8.5.3 To what extent do brokers facilitate the development of transactive memory?

Previous studies suggest that creating clusters in a GSD environment inhibits the development of transactive memory. A common approach to eliminate those challenges is the role of brokers. We therefore examine the hypothesis that brokers moderate the negative effect that clustering has on transactive memory (Fig. 8.3).

Figure 8.3: Hypothesis: Brokers as moderators

To test the hypothesis and answer our last question, we performed a regression analysis using the following variables:

- **Dependent variable:** The individual scores of transactive memory calculated using Eq. 7.2 (chapter 7).
- **Independent variable:** The individual clustering coefficient scores that we described in Section 8.4.2
- **Independent variable:** The betweenness centrality scores that we use to represent brokerage.
• **Control variable**: The role of team leader. This is a dichotomous variable indicating whether a member is a team leader or not.

• **Control variable**: The role as code contributor. This is also a dichotomous variable indicating whether a member contributed code commits, or not.

A common problem in multiple regression models, and especially in moderation is multicollinearity [134]. Multicollinearity exists when there is high correlation between the independent variables. A popular remedy to multicollinearity is to center the independent variables at their mean [38, 91]. We use this approach to transform the independent variables of our moderation model (betweenness and clustering coefficient). Next, we tested the variance inflation factors (VIF), a measure for estimating the significance of the correlation between the independent variables. All VIFs have values lower than 10 which is the threshold suggested by [134]. Therefore, we can conclude that there is no significant multicollinearity between the independent variables of our model.

Table 8.3 presents the results of the analysis. Model 1 shows that the two control variables of leadership role ($\beta = .135, p > .05$) and code contribution ($\beta = .255, p > .05$), do not have any significant effect on transactive memory. In Model 2, betweenness has a significant positive effect on TM ($\beta = .350, p < .05$), and clustering has a significant negative effect ($\beta = -.606, p < .001$). Model 3 includes the interaction of betweenness and clustering (interaction\_BxC), representing the moderation effect and it has a significant positive effect on TM ($\beta = 1.118, p < .05$). These results validate our hypothesis that brokerage moderates the effect of clustering on transactive memory.

**Table 8.3: Results of Regression Analysis on Transactive Memory**

<table>
<thead>
<tr>
<th></th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
<th>Descriptives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$B$</td>
<td>$SE$</td>
<td>$\beta$</td>
<td>$B$</td>
</tr>
<tr>
<td><strong>Control Variables</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>.218</td>
<td>.286</td>
<td>.135</td>
<td>-.250</td>
</tr>
<tr>
<td>Code Contributor</td>
<td>.313</td>
<td>.217</td>
<td>.255</td>
<td>-.134</td>
</tr>
<tr>
<td><strong>Independent Variables</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Betweenness</td>
<td>.011</td>
<td>.004</td>
<td>.350*</td>
<td>.058</td>
</tr>
<tr>
<td>Clustering</td>
<td>-1.875</td>
<td>.423</td>
<td>-.606**</td>
<td>.974</td>
</tr>
<tr>
<td>interaction_BxC</td>
<td>0.168</td>
<td>.053</td>
<td></td>
<td>1.118*</td>
</tr>
<tr>
<td>Adj-$R^2$</td>
<td>.004</td>
<td>.647</td>
<td>.726</td>
<td></td>
</tr>
<tr>
<td>$\Delta R^2$</td>
<td>.060</td>
<td>.627**</td>
<td>.077**</td>
<td></td>
</tr>
<tr>
<td>F-statistics</td>
<td>1.08</td>
<td>17.49**</td>
<td>20.03**</td>
<td></td>
</tr>
</tbody>
</table>

$^*p < .05, \quad ^{* *}p < .001$

To further support our conclusions, we apply the PROCESS, a method proposed
8.6. Threats to validity

by Hayes [75]. PROCESS is a freely-available regression-based path analysis (for SPSS and SAS) that estimates the model coefficients in mediation and moderation [75]. Using PROCESS, we can identify how brokerage (measured through betweenness) moderates the effect that clustering has on transactive memory. The results are presented in Table 8.4. We notice that for low scores of betweenness (one standard deviation below the mean) there is a significant negative effect of clustering on transactive memory ($\beta = -1.20$, $p < .001$). At the mean levels of betweenness, the effect is positive, although not significant ($\beta = .89$, $p > .05$). At high scores of betweenness (one standard deviation above the mean), the effect is significantly positive ($\beta = 3.88$, $p < .05$). In other words, we find that among clustered members, those who also act as brokers (i.e. those with high betweenness scores) will be better aware of where expertise resides within the network, how to access, share and store expertise knowledge, than those who do not act as brokers (i.e. those with lower betweenness scores).

Table 8.4: Moderation effect on Transactive Memory

<table>
<thead>
<tr>
<th>Betweenness</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>.00</td>
<td>-1.20** ( .42 )</td>
</tr>
<tr>
<td>13.08</td>
<td>.89 ( .98 )</td>
</tr>
<tr>
<td>31.77</td>
<td>3.88* ( 1.95 )</td>
</tr>
</tbody>
</table>

Bringing everything together, we conclude that brokerage moderates the negative effect that clustering has on the development of transactive memory. Our results suggest that members who are more often “in-between” many others, within a clustered network, will overcome the knowledge management challenges brought by the clusters and they will develop better transactive memory.

8.6 Threats to validity

The credibility of a study refers to the degree of confidence we have on its conclusions [150]. Although we have addressed the validity of our study from the beginning of this research, it is important to point out several aspects that might be a threat. We do so for two reasons; first because we want to establish the integrity and credibility of our observations and results, and second for suggesting
CHAPTER 8. THE ROLE OF BROKERS AS FACILITATORS OF TMS

future improvements.

To tackle internal validity in our study, we carefully collected our social network data and achieved a 92% response rate for an accurate representation of the collaboration network. Nevertheless, we do not take into account certain individual attributes of the participants, such as the background knowledge and experience. For instance, academic partners often participate in EU-funded projects where their main focus is on research, rather than on the technical contribution. Cultural aspects might also influence people’s responses. In this paper, we do not take these aspects into account but we suggest their consideration for the future. Additionally, real networks are not static, but rather dynamic. Relationships between people change and as such, communication networks evolve. In the current study, we examine only a snapshot of the networks. An interesting approach for a subsequent study is to compare the development of transactive memory over time and observe networks’ evolution.

While we report on the reliability analysis for the latent variable model of TM and the multicollinearity tests of the regression model, we recognize that there might be further threats to construct validity. For instance, identifying the domain expertise of every member in the network (i.e. the type of specialized knowledge that each possess) can improve the operationalization model of TM. We propose therefore that the theoretical concepts and the variables used in the operationalization of the model be further re-evaluated, tested, validated and potentially improved.

Finally, there might be factors posing a threat to external validity for our case study. Although, we discuss the differences that an EU-funded project might have compared to commercial or OSS projects, our observations might be influenced by particular and distinctive characteristics of the partners. For instance, some partners might have experience working together in other projects, and consequently their collaboration might differ from partners who work together for the first time. Also, EU projects’ life time might vary from one to approximately four years, which also influences how well team members get to know each other. Whereas “a theory may never be generalized to a setting where it has not yet been empirically tested and confirmed” [108], we suggest that using additional case studies to the research can eliminate the threat to external validity.

8.7 Conclusion

With the evergrowing body of knowledge management challenges in global software development, solutions are of paramount importance. In this paper, we
8.7. CONCLUSION

present the role of brokers as a solution to overcome challenges in GSD, and particularly to facilitate the development of transactive memory. TM is of special interest in knowledge-driven projects due to the specific focus on expertise knowledge that team members possess, rather than general knowledge that everybody shares. Particularly, we used social network data collected from a multi-site, EU-funded research project and we have answered three research questions:

1. **What kind of clusters emerge in the project’s collaboration structure?** The results show that in our case study, there is a moderate tendency for task-based clustering.

2. **Which members play the role of brokers and what are their characteristics?** We find that brokers (designated or emergent) are members responsible for the overall coordination of the project, as well as members with high technical contributions.

3. **To what extent do brokers facilitate the development of transactive memory?** Our empirical evaluation shows that clustering has a negative effect on TM and that brokers can moderate that effect.

From a practical point of view, we provide an analytical method to identify clusters created within multi-site collaboration networks, as well as the brokers that emerge between those clusters. Considering the importance of brokers in overcoming knowledge management challenges, practitioners may benefit from our results to foster this particular role. Likewise, teams can analyse their structure and composition and decide whether to assign formal brokers or encourage the emergence of informal brokers.

From a theoretical perspective, we add to the body of knowledge management challenges in GSD by presenting the role of brokers as facilitators on the development of transactive memory. As GSD research evolves and the attention to knowledge management challenges increases, solutions such as using brokers can benefit future studies. Additionally, we analyse an EU-funded project, which is a multi-site software development project with mainly research rather than commercial interests. Up to date, research in GSD focus either on commercial and proprietary software, or on OSS. As Prause et. al. note [154], these research projects can be considered a third type of GSD projects besides industrial and open source and they exhibit different characteristics. In this study for instance, we found that members of a research-driven project tend to create task-based clusters, rather than location-based clusters. We also observed that there can be both formal brokers with leadership positions (a characteristic commonly found in industrial cases with appointed team leaders), as well as informal brokers defined based on their technical contribution (a characteristic commonly found in OSS projects).
Transactive Memory before and after Software Transfers

In this chapter, we present the results of a longitudinal case study in multi-site software governance. We examine the changes implemented over the past two years in the governance of an international company where software development activities and responsibilities were transferred between remote locations. We compare the governance model before and after the transfer. Particularly, we use the theory of transactive memory to evaluate the effect that the governance changes brought in expertise awareness, accessibility, credibility and communication frequency of the dispersed team members.

9.1 Introduction

When companies engage into global software development, they may have to put some effort in transferring development activities from one location to another. Such efforts are associated with several knowledge transfer challenges, which in turn may result in decreased productivity [26]. For instance, an important challenge is to transfer the competence, i.e. to make sure that the new teams have the right resources and skills to continue the work after the transfer is complete [168]. In the same vein, the receiving teams must become familiar with the new tasks, learn new skills and “deal with the implicit knowledge” [168]. Consequently, communicating and managing knowledge is a key aspect in software transfers.

Moreover, transferring software development from one location to another leads to different governance decisions that need to be made with regard to how tasks will be allocated and how the new teams will be staffed. Mockus and Weiss [128], for
instance, argue that tightly coupled work items that require frequent coordination and synchronization should be performed within one site. This observation refers to the component-based division of work between remote locations, where one team is responsible for the development of a single component, and another team is responsible for another component. In this way, interdependencies between remote teams are limited and hence, GSD collaboration is facilitated. On the other hand, Kotlarsky et. al [101] found that an expertise-based division of work can also prove beneficial, allowing the GSD companies to access the pool of expertise available in offshore locations. An expertise-based task allocation implies that the development work is distributed between the remote sites based on the functional and technical expertise available on each site.

In this chapter, we examine the changes implemented over the past two years in the governance of an international company, where software development activities and responsibilities were transferred between remote locations. We compare the governance model before and after the transfer. Particularly, we use the theory of transactive memory to evaluate the effect that the governance changes brought in expertise awareness, accessibility, credibility and communication frequency of the dispersed team members.

9.2 Software transfers and governance

Software transfer is defined as “a project where work is moved from one development site to another development site.” [185]. In that regard, software transfers are closely related to the different governance decisions that one should consider when involved in global software development activities. As we described in chapter 2, in this book we look at governance from three aspects; the business strategy that binds the relationship of the remote offices, the structure and composition of the remote teams and the way tasks are allocated across sites.

Software transfers may be influenced by the business strategy between the onshore and the offshore offices. For example, when companies start a new collaboration with an external partner (offshore outsourcing) only part of the development activities are transferred [20]. On the other hand, when collaboration between remote locations is based on an offshore insource business strategy, more responsibilities and development activities may be transferred to the remote offices [122].

Additionally, software transfers are associated with deciding how development tasks will be allocated across sites. Many GSD companies, for instance, transfer only part of the development activities to offshore partners. Some of the tasks
that are more likely to remain on-site are those related to the core capabilities of
the software, or other activities that require customer interaction and therefore
customer proximity [26]. In another example, in chapter 7, we discussed how Ri-
con company employs what they call “steering strategies”, which define how tasks
will be allocated and how much will be transferred to the remote offices. These
steering strategies depend on the strategic importance and/or the complexity of
the development work. Managers decide how much of the responsibilities will be
delegated to the external partners, ranging from having a team executing only
software development tasks, to a self-sustaining and self-managing team.

Following the above case, software transfers involve also changes in the composi-
tion and the structure of the teams. When new tasks and activities are transferred
from one location to another, new teams may be created or new members may
be introduced to the existing teams. For instance, when a company decides to
transfer architectural decision-making to the remote teams, a team member will
be appointed to fulfill the new role of architect. The importance of staffing dur-
ing a software transfer is mainly recognized in [185]. The authors used staffing
curves to represent how teams and members from remote locations change during
a software transfer. Based on their observations, the authors were able to com-
pare two different types of software transfer; a full transfer, where the teams at
the receiving site took over the development of a complete project, and a partial
transfer, where tasks were gradually transferred and hence, staffing of the teams
was piecemeal.

Thus, software transfer projects result in changes of the GSD governance model.
In order to relocate development work from one development site to another,
decisions need to be taken with regard to how tasks will be transferred and
allocated between geographically dispersed members. Finally, managers should
decide how the new or existing teams will be structured in order to secure an
effective software transfer.

9.3 Knowledge transfers and transactive memory

Transferring software development activities from one location to another entails
the transfer of the related knowledge to perform those activities [100]. Knowledge
transfer is a process through which one organization identifies and learns specific
knowledge that resides in another organization, and reapply this knowledge
[73]. One of the most challenging types of knowledge to be transferred in global
software development is the domain knowledge, i.e. the specialized knowledge
embedded in people’s minds and organizational processes and routines [26] [138].
CHAPTER 9. TRANSACTIVE MEMORY BEFORE AND AFTER SOFTWARE TRANSFERS

Differences in skills, expertise, technical infrastructure and methodologies hinder domain knowledge transfer between remote sites.

Oshri et. al [144] proposed that transactive memory systems can support knowledge transfer in global software development and facilitate the coordination of collective expertise. In chapter 9, we introduced transactive memory system as a shared cognitive system for encoding, storing, and retrieving knowledge between members of a group. We further presented an operationalizational model to measure transactive memory processes in GSD. Our proposed model is based on (1) the three basic processes of a transactive memory system: directory updating, information allocation and retrieval coordination [152] and (2) on four main aspects that influence the development of transactive memory; expertise awareness, accessibility, credibility and communication frequency.

Furthermore, transactive memory systems are an evolving rather than a static phenomenon [21]. Every work group develops a transactive memory system to some extent by spending time together and learning how expertise is distributed within the group [131]. Additionally, job rotation is argued to have an effect on transactive memory due to frequent changes in group membership, which may lead to incorrect and incomplete expertise awareness [132]. Transactive memory therefore evolves over time due to several contextual factors such as the maturation of the relationship between members working together, the dynamic task assignments, or job rotations.

In this chapter, we analyze transactive memory in a longitudinal case study where software transfers were implemented over the course of two years. Specifically, we examine how expertise awareness, accessibility, credibility and communication frequency changed before and after the software transfers.

9.4 The case study

In this chapter, we revisit the case study presented in chapter 5. During that case study, we collected data from a multi-site software product line in two different points in time and we compared the changes in the organizational structure of the SPL. In this chapter, we extend that work by analyzing the changes in the governance model of the SPL as a result of the software transfers. We also identify the effect that those changes have on the development of transactive memory.

As we described in chapter 5, we used an online survey to collect the data. We distributed the survey to all project members and invited the respondents to identify those with whom they mainly collaborate. Additionally, we asked the respondents to value the relationship (on a 5-point Likert scale) with each one
of their connections, based on the questions Q1-Q6 presented in chapter 6. The end result is a set of data that represents who communicates with whom, and a set of six valued relationships for every connection.

Furthermore, we performed 10 semi-structured interviews with members from both site NL and site A. Table 9.1 presents in details the roles of the interviewees and the number of interviews performed in each site. During the interviews, we used a protocol to guide the discussions [189]. The protocol included two main parts; the first part was focused on the software transfers and organizational changes over the past two years. The second part aimed at collecting opinions of the interviewees with regard to the aforementioned changes and the effect that these changes brought. All interviews were recorded and transcribed. To process the data, the Atlas.ti tool was used, a commercial software for qualitative analysis of textual and visual data. Finally, we had several focus groups with contact people from the company, during which we discussed details of our research, presented our findings and interpreted the results.

Table 9.1: Interviewees Lists

<table>
<thead>
<tr>
<th>Role</th>
<th>site NL</th>
<th>site A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product leaders</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Team leaders</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Architects</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>Integrators</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total interviews</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

In the next section, we present the software transfers realized within the period of two years (2011-2013). We use the governance model presented in chapter 2 to analyze the changes in the organization of the GSD activities, as a result of the software transfers.

9.5 Changes in governance model

From an organizational perspective, site NL and site A are part of the same research and development department of Océ. Following the taxonomy in [169], the defined business strategy is offshore insource, with near geographic distance and small temporal distance. Consequently, the remote offices are allowed to freely exchange knowledge and the distributed colleagues collaborate with each

1www.atlasti.com
other without any barriers limiting their communication. The business strategy between the two locations was not affected by the software transfers.

The first change in the governance model relates to task allocation as software development activities are transferred from one location to another. Specifically by November 2011, which is the beginning of our research timeline, the development of the software components is distributed between teams in site NL and site A, i.e. a component is either the responsibility of a team in the Netherlands, or a team at site A. In January 2012, it was decided to transfer the development of all software components to site A. This software transfer did not have an effect in teams’ organization. In fact, the development work transferred from site NL was received by existing teams in site A.

In September 2013, the teams in site A were re-organized from component-focused to product-focused. Instead of having separate teams developing the controller components, now each team is responsible for a product within the product line and the team members work on all components within that product. Additionally, they migrated from a traditional waterfall development methodology to an agile way of working using Scrum. Looking at the governance model, these decisions resulted in a change in teams’ structure and composition as new teams were created and new roles were introduced.

A final change in the governance model of the SPL relates to an additional software transfer from site NL to site A. In November 2013, it was decided to transfer the product integration activities from site NL to site A. Now, teams in site A develop and deliver to site NL a complete, integrated product package. As a result, new members were introduced to fulfill the role of product integrator.

Table 9.2 presents all three changes in the GSD governance for the period 2011-2013.

9.6 Changes in transactive memory

As we described in chapter 6, the questions we use to measure transactive memory processes represent four basic manifestations of transactive memory:

1. **Expertise awareness** represents the extent to which members of the team are aware of the expertise domain of their colleagues. To measure expertise awareness we calculate the average of questions Q1 and Q2.

2. **Accessibility** represents how easy (or difficult) it is to access the right persons within the network. Accessibility is measured using question Q4.
Table 9.2: Changes in the governance model

<table>
<thead>
<tr>
<th>Changes in Governance model</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>changes in Task Allocation</td>
<td>The development of software components is distributed between site NL and site A</td>
<td>The development of software components is in site A.</td>
</tr>
<tr>
<td>changes in Team Structure &amp; Composition</td>
<td>Distributed teams are component-focused. In site NL, there are people at the product level combining all components.</td>
<td>All teams in site A are product-focused and they adopted the Scrum methodology. The team delivers the different components to site NL for product integration and testing.</td>
</tr>
<tr>
<td>changes in Task Allocation and Team Structure &amp; Composition</td>
<td>Product integration is performed in site NL. The product integrator collects different components developed from different teams.</td>
<td>Product integration is performed in site A. The product integrator is part of the product team in site A and the team delivers a complete, integrated product package.</td>
</tr>
</tbody>
</table>

3. Credibility represents the extent to which members trust the information they receive from their colleagues. Credibility is measured using question Q5.

4. Communication frequency refers to how often people come in contact with their colleagues to retrieve or allocate information related to their expertise. To measure communication frequency we calculate the average of questions Q3 and Q6.

The purpose in this chapter is to see how transactive memory was affected by the changes in the governance, over time. We therefore begin the analysis by comparing the results between 2011 and 2013. Table 9.3 presents the descriptive statistics of the data and the results from the t-test. We observe that in all transactive memory variables, the scores of 2013 are lower than the scores of 2011. To validate these observations, we perform a t-test analysis and compare the means of the two data sets. The results show that the mean scores of expertise awareness, accessibility, credibility and communication frequency in 2013 are significantly lower than the mean scores of 2011.

Furthermore, we performed the same test taking into consideration the people that were in the SPL before and after the software transfers. Particularly, we find the 44 people were part of the SPL in 2011 and 2013, i.e. approximately 50% of the members are the same before and after the changes. Table 9.4 present the results.

In order to corroborate the results from the online survey, we analyze the interviews and identify the reasons supporting and further explaining the decrease of the scores. The interviews were analyzed based on the coding process of micro-
CHAPTER 9. TRANSACTIVE MEMORY BEFORE AND AFTER SOFTWARE TRANSFERS

Table 9.3: Descriptive Statistics & t-Test results

<table>
<thead>
<tr>
<th></th>
<th>2011 mean</th>
<th>SD</th>
<th>2013 mean</th>
<th>SD</th>
<th>t-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expertise Awareness</td>
<td>4.05</td>
<td>.59</td>
<td>2.14</td>
<td>.38</td>
<td>25.59**</td>
</tr>
<tr>
<td>Accessibility</td>
<td>4.47</td>
<td>.63</td>
<td>2.34</td>
<td>.47</td>
<td>26.10**</td>
</tr>
<tr>
<td>Credibility</td>
<td>4.23</td>
<td>.56</td>
<td>2.26</td>
<td>.44</td>
<td>26.67**</td>
</tr>
<tr>
<td>Communication Frequency</td>
<td>2.54</td>
<td>.44</td>
<td>1.86</td>
<td>.39</td>
<td>11.37**</td>
</tr>
</tbody>
</table>

** sign. at p < .001

Table 9.4: Descriptive Statistics for common members & t-Test results

<table>
<thead>
<tr>
<th></th>
<th>2011 mean</th>
<th>SD</th>
<th>2013 mean</th>
<th>SD</th>
<th>t-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expertise Awareness</td>
<td>4.09</td>
<td>.50</td>
<td>2.13</td>
<td>.41</td>
<td>20.11**</td>
</tr>
<tr>
<td>Accessibility</td>
<td>4.49</td>
<td>.56</td>
<td>2.36</td>
<td>.49</td>
<td>18.87**</td>
</tr>
<tr>
<td>Credibility</td>
<td>4.39</td>
<td>.51</td>
<td>2.22</td>
<td>.43</td>
<td>21.33**</td>
</tr>
<tr>
<td>Communication Frequency</td>
<td>2.52</td>
<td>.34</td>
<td>1.86</td>
<td>.35</td>
<td>8.44**</td>
</tr>
</tbody>
</table>

** sign. at p < .001

Analysis or otherwise called a “line-by-line” analysis [174]. The codes were eventually grouped into categories depending whether they reflect expertise awareness, accessibility, credibility or communication frequency. The following sections present the results of the qualitative analysis.

9.6.1 Expertise awareness

An important reason that contributed to the decrease in expertise awareness is that a lot of people with expertise in a certain domain, or a specific task changed projects. Following the transfer of software activities from site NL to site A, members in site NL that worked on specific components or parts of the system changed projects:

“For example, in the X project they had one guy who was working for years and he is now being replaced” (Team Leader, site A)

Additionally, while people changed projects they had to learn new skills and perform new tasks. Consequently, they are not yet ‘experts’ in their new field:

“They also have to move to different software technology, so a lot of C++ programmers suddenly they have to learn java. Maybe that’s
9.6. CHANGES IN TRANSACTIVE MEMORY

“not a huge step, but it’s also problematic.” (Architect, site NL)

Finally, the transition from component-focused teams to product-focused teams implies that now there is a pool of developers and depending on the priorities in the product backlog, tasks will be assigned to team members. This approach hinders the development of specialization and expertise:

“The idea is that you have team capable of doing everything, so you do not have specialised people. The difficulty then is that you need to be specialist to run applications, you need to be a specialist to do for example User Interface work, and it takes years to learn. So the idea as it is now is that you do not have specialised teams” (Team Leader, site A)

9.6.2 Accessibility

Analyzing accessibility aspects from the interviews, we find that an important factor is that people are appointed to new positions, and the previous ones are not accessible anymore as they work in other tasks:

“But now we can see for this component for example, if we need something from the old software component or implementation, then we will have trouble.” (Team Leader, site A)

Furthermore, the transition and adoption of Scrum methodology in site A is not yet settled. Consequently, the roles of Scrum Master and Product owner are not clearly defined and the people from site NL have difficulties accessing the right person:

“There is still a major discussion between site NL and site A, because site A wants to call this guy the Product Owner and site NL says no that is not a Product Owner.” (Integrator, site NL)

Also, due to the recent re-organization of the teams in site A and the on-going transfer of integration activities, it is not yet clear who is in which team and who works on which part of the product:

“It’s one big team and everybody is working on everything, so currently I’m not sure if I know who to contact. I’m not sure if I have a problem that I will find the right people.” (Architect, site NL)
CHAPTER 9. TRANSACTIVE MEMORY BEFORE AND AFTER SOFTWARE TRANSFERS

9.6.3 Credibility

Accuracy and credibility of the shared knowledge between members of a network develops over time [148]. Littlepage et. al [118] suggest that working together in the network provides opportunity to further develop accurate perceptions of other’s expertise and trust more the information received. Our findings show that software transfers and teams’ restructuring lead to a decrease in credibility. Analyzing the interviews, we found that other people do not always trust the information they receive from their newly-appointed colleagues:

“I’m not supposed to ask the one that has the knowledge, but I am supposed to ask the one that has the responsibility now, and not the knowledge. And sometimes I do both.” (Architect, site NL)

9.6.4 Communication frequency

The transfer of all development activities from site NL to site A, as well as the most recent transfer of product integration activities give the opportunity to the teams in site A to be more independent and take more responsibilities. Consequently, the communication between the two locations decreased:

“At some point you needed to discuss with 5 or 6 guys from both sites, in order to have some alignment. Now we are only three people in site A looking at the subject.” (Team Leader, site A)

Additionally, following the new arrangements of product-focused teams, the communication between site NL and site A is the responsibility of fewer, designated members who act as bridges between locations:

“So now we have an interface between site A and site NL and some specific people dedicated to that interface.” (Product Leader, site NL)

9.7 Discussion

In the previous sections, we have presented a quantitative and a qualitative analysis of transactive memory before and after software transfers. The quantitative analysis of TM revealed that transactive memory has significantly decreased in 2013, compared to the scores obtained in 2011. To further explore why transactive memory decreased, we analyzed the interviews and identified how software transfers influenced expertise awareness, accessibility, credibility and communication frequency. Table 9.5 summarizes the results.
## Table 9.5: The effect of software transfers in TMS

<table>
<thead>
<tr>
<th>Transactive Memory aspects</th>
<th>How software transfers influenced TM</th>
</tr>
</thead>
</table>
| Expertise Awareness        | - People with experience in a specific field changed projects.  
                              - People have to learn new skills to accomplish their new tasks.  
                              - The transition to product-focused teams hinders specialization and expertise among team members. |
| Accessibility              | - People that have the knowledge are not accessible anymore because they work on different tasks.  
                              - Roles and responsibilities are not clear yet.  
                              - It is not clear who works on which team. |
| Credibility                | - People do not always trust the information they receive from their newly-appointed colleagues. |
| Communication Frequency    | - The teams in site A are more independent.  
                              - There are fewer, designated people for the communication between locations. |

Software transfers resulted in a decrease in expertise awareness due to three reasons. First, members with experience in specific domain changed projects and consequently people were not aware anymore who has the needed knowledge. Second, people had to acquire new knowledge and learn new skills to perform their new tasks, and consequently they were not “experts” anymore. Third, it is argued that the product-focused organization of the teams hinders specialization, as anyone is free to work in any part of the product.

Moreover, the results showed that members of the teams could access their colleagues easier before the transfers. Three reasons were identified that provide support to the decreased accessibility: first, members that have developed a particular expertise in a field were not available anymore since they have changed teams and projects. Second, our research was conducted in a period where the situation was in a state of flux and as a result the new roles and responsibilities between the teams in site A and site NL were not yet clear. The latter observation leads to a decrease in accessibility as people do not know who is responsible for which part of the project, and to whom should they refer when information is needed.

Credibility was also affected by the software transfers. The main reason identified through the interviews was that people could not easily trust the information they received, because expertise was “lost” while transferring development work and knowledge from one site to another. As Jabangwe and Smite [90] note “a transfer disrupts the continuity of the knowledge”.

Finally, communication frequency decreased after the transfer due to two reasons. First, teams in site A became responsible for the development work of all the product components. Following this, the teams in site A are now more
independent from their colleagues in site NL in terms of decision-making and hence, the need to communicate frequently is diminished. Second, with the new organization designated people were appointed responsible for the communication between sites. The result is that the communication lines between locations are now limited to those designated members, and consequently the overall communication frequency has decreased.

Summarizing, our results show that due to the software transfers and the resulted changes in GSD governance, the transactive memory has decreased over time. Our observations support previous studies that argue that software transfers may affect GSD performance [90, 128]. Although we have not directly measured performance and productivity in our case study, we support the decrease in efficiency by observing the decrease in TMS. As one of the interviewees said:

“At this moment, people mainly focus on acquiring knowledge which they don’t have yet, and not giving something back.” (ex-Team leader, site NL)

Currently, we find that there is not enough information at hand to “predict” the future benefits realized by the transfers. This can be accomplished with a follow-up case study, when all software transfers would be completed and all organizational changes will settle.

9.8 Conclusion

In this chapter, we presented the results of a longitudinal case study in multi-site software governance. We examined the software transfers implemented over the course of two years and the resulting changes in governance of a software product line. We compared the governance model before and after the transfers. Particularly, we used the theory of transactive memory to evaluate the effect that the software transfers brought in expertise awareness, accessibility, credibility and communication frequency of the dispersed team members.

The contribution of this final case study is twofold. First, we examine software transfers as an aspect that influences the GSD governance model of a company. Our results support previous studies suggesting that software transfers need to be carefully managed because they may lead to a disruption or loss of expertise knowledge, and hence to communication and collaboration hurdles. Second, we analyze transactive memory based on four aspects that influence its development, and not as a single, latent variable. This approach provides a more analytical view, an “anatomy” of transactive memory within global software development.
We also present the dynamic nature of transactive memory through a longitudinal case study. As Brandon and Hollingshead \cite{21} note “because TMS involves processes that unfold over time, studies that measure TMS at multiple points in time are imperative".
Conclusion

This thesis focused on the effect of global software governance on transactive memory systems by taking a social network perspective. Particularly, we proposed a model to outline the different decisions taken in order to manage GSD activities. Managing such activities also influences the way geographically dispersed teams collaborate. To that end, we used social network analysis for identifying different collaboration structures. Additionally, working over distance inhibits the development of transactive memory systems. We proposed a method in order to capture and analyze such systems in GSD. In this final chapter, we revisit the research questions introduced in chapter 1 and we summarize the answers provided throughout the thesis. We also reflect on the overall contributions of our work and conclude with suggestions on how the present thesis can be extended and further supported by future research.

10.1 Answering the research questions

In this thesis, we studied the effect of global software governance on transactive memory systems by taking a social network perspective. In chapter 1, we decomposed the main research question (How does global software governance affect transactive memory systems?) into three sub-questions. Throughout the thesis, we provided an answer to those questions. In the next paragraphs, we summarize our findings:
CHAPTER 10. CONCLUSION

10.1.1 RQ.1 - How is global software governance defined?

Globalization of software development activities increases the challenges associated with managing and controlling geographically dispersed teams. In chapter 2, we introduced a multi-site software governance model, that helped us identify and categorize governance decisions. Particularly, we proposed a model based on three aspects; the business strategy that binds the relationship of the remote offices, the structure and composition of the remote teams and the way tasks are allocated across sites. With this model, we provide an answer to question RQ.1.1 What aspects embody a global software governance model?

Additionally, we answer the second sub-question RQ.1.2 What are the implications of governance decisions in global software development? We performed several case studies, where we used the multi-site governance model to classify GSD activities and decisions. The purpose was to examine how different decisions affect the collaboration between the remote offices. In chapter 1, we examined the effect that different governance decisions have on several knowledge management challenges and we found, for instance, that when there are legal barriers between remote offices the information transferred from one location to another needs to be filtered. Consequently, there are delays and misunderstandings in the collaboration. In chapter 2, we applied the governance model to capture the transition from a traditional software development methodology (RUP), to an agile way of working using Scrum. The results showed that a migration from RUP to Scrum brings a positive effect in requirements engineering, cost management and cross-functionality of the distributed teams.

In chapter 7, we performed a comparative case study between four GSD projects. We used the governance model to classify their activities and we reported on the effect that different governance decisions have on transactive memory systems. We found that, for instance, in some projects there was a dedicated team acting as the interface between remote locations. The members of this “team-in-between” were better aware of who knows what within and between locations, they could easily access remote and co-located colleagues and overall, they had better developed transactive memory processes. Moreover, we presented a novel mapping between task allocation and the composition of the core members.

Finally, in chapter 9 we examined the changes in the governance model of a multi-site software product line, which resulted from software transfers between two remote locations. Our results showed that software transfers affected the governance decisions in terms of task allocation and team structure and composition.

Bringing everything together, we answered RQ.1 by proposing a multi-site gover-
10.1. ANSWERING THE RESEARCH QUESTIONS

nance model to define the main aspects that need to be considered when managing global software development activities. We also applied the model to several case studies and we reported on the implications that different governance decisions have in global software development.

10.1.2 RQ.2 - How can we identify and analyze collaboration structures in global software development?

In the past few years, social network analysis (SNA) techniques are used to understand collaboration patterns within organizations and reveal collaboration structures that go beyond the organizational charts and the project planning. In this thesis, we posed question RQ.2.1 *How can we use social network analysis to explore GSD collaboration?* In chapter 3, we performed a systematic literature review, where we collected and analyzed previous work that adopt a social network perspective in distributed software development. We found that SNA techniques can be used to identify the structure of the network, such as core-periphery structures and clusters. Core-peripheries are created when a particular group of people are more active in participating and coordinating the group activities while the rest of the members are in the periphery. Clusters, on the other hand, are created when certain common characteristics (e.g. geographic location, shared tasks and assignments, common interests, etc.) bring people closer to each other, forming subgroups. Existing literature also suggests the use of SNA for revealing the important and influential members of the network.

Moreover, we answered sub-question RQ.2.2 (*What kind of collaboration structures exist?*) by using the results from the SLR and applying SNA techniques in our case studies. Specifically, in chapter 5 we analyzed the collaboration structures in a multi-site software product line while there was a transition from component-focused teams to product-focused teams. Using social network analysis, we found that the resulted collaboration structure of the networks changed from centralized to decentralized organization. Additionally, in chapter 8 we used SNA techniques to identify the brokers within a multi-site, research project and we showed how those brokers can moderate the effect of clustering in TMS. Furthermore, in the same chapter we described two different types of clusters that exist in literature: task-based and location-based. Task-based clusters are more common in OSS projects whereas location-based clusters are more common in commercial GSD projects.

The combined effort towards answering RQ.2.1 and RQ.2.2 provides a comprehensive response to RQ.2; we can use social network analysis as a technique to identify and explore collaboration structures in GSD, by analyzing the tendency
the members in the network have to create clusters, what kind of characteristics bring together those clusters, and how well connected they are. We can also use SNA to identify important and central members in the network such as core members and boundary spanners (or brokers). Due to their position within the network these roles facilitate collaboration in global software development.

10.1.3 RQ.3 - How are transactive memory systems defined in GSD?

Working over distance inhibits team members from knowing “who knows what”. In order to capture and analyze the extent to which distributed team members are aware of each others’ expertise, how accessible they are and how often they communicate, we proposed a method to measure transactive memory systems and answered sub-question RQ.3.1 (How can we identify and measure transactive memory systems?).

According to Wegner et al. [183], transactive memory has two components: an organized store of knowledge (TM structure) and a set of transactions/processes (TM processes) related to knowledge management processes of encoding, storing, and retrieving. The TM structure is a knowledge representation of members’ shared understanding of who knows what. TM processes are the mechanisms by which the group coordinates members’ learning and retrieval of knowledge, so that the knowledge can be applied to group tasks. In this thesis, we examined both the structure of transactive memory systems and the processes involved.

In order to identify and examine the structure of transactive memory systems, we adopted a social network approach. Particularly, we proposed the use of three common SNA approaches to analyze transactive memory structures: Core-peripheries, clusters and centrality measures. Furthermore, we presented a latent variable model for measuring the transactive memory processes. According to Lewis [115], a measure of TMS must meet two basic criteria; the first is to be theoretically consistent with Wegner’s conceptualization of TMS that reflect the cooperative processes and transactions of memory used in a group [182]. The second criterion is to align the measurements with the settings of the field we are interested in studying. Following these suggestions, we considered previous studies and conceptualizations of the theory around TMS and we created a latent variable model to measure transactive memory processes in global software development.

In chapter 7, we applied our proposed method in order to measure transactive memory systems in four different GSD projects. Answering sub-question RQ.3.2 (What aspects influence the development of transactive memory systems?), we
also examined how those systems were affected by the different governance decisions followed in each project. Our results suggest that business strategy decisions such as pursuing an offshore insource approach, creates tightly-connected clusters, which in turn leads to better developed transactive memory systems. We also observed that within the composition and structure of GSD teams, there are boundary spanners who have a better overview of the network’s activities and bring remote teams together by facilitating knowledge sharing. Due to these attributes that boundary spanners have, they become central members within their network and they develop better transactive memory processes.

In chapter 9, we analyzed transactive memory based on four aspects that influence its development namely, expertise awareness, accessibility, credibility and communication frequency. The results showed that several governance changes that were implemented over a period of two years, lead to a decrease in transactive memory. The latter case study contributes to the results of the current thesis, by presenting the dynamic nature of transactive memory systems.

Summarizing our findings, we were able to provide an answer to question RQ.3 by proposing a method to measure transactive memory systems in GSD. We applied this method to several projects and we reported our empirical observations on how governance decisions affect the development of TMS in global software development.

10.1.4 Main Research Question - How does global software governance affect transactive memory systems?

The collective effort to explore each of these aforementioned sub-questions provides an answer to our main research question. We answered the main research questions in three steps; first we defined what is governance, second we defined transactive memory systems, and third we explored through several case studies the effect of governance on TMS, by taking a social network perspective.

To answer the main question we first present a systematic way of analyzing governance structures in GSD and their effect in transactive memory systems. Particularly, we suggest the use of the GSD governance model to analyze multi-site activities based on three aspects namely, the business strategy, task allocation, and team structure and composition. We also apply a twofold approach to analyze transactive memory systems; a process and a structural approach.

Transactive memory processes and structures are affected by business strategy decisions, dynamic task assignments and flexible teams. For instance, we have seen that an offshore insource business strategy results in stronger transactive
Figure 10.1: Answering the main research question

<table>
<thead>
<tr>
<th>GLOBAL SOFTWARE GOVERNANCE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined based on:</td>
<td></td>
</tr>
<tr>
<td>- Business Strategy (e.g. offshore insource, offshore outsource)</td>
<td></td>
</tr>
<tr>
<td>- Task Allocation</td>
<td></td>
</tr>
<tr>
<td>- Team Structure &amp; composition</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRANSACTIVE MEMORY SYSTEMS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TM processes</td>
<td>TM structure</td>
</tr>
<tr>
<td>A latent variable model</td>
<td>Applying SNA to identify:</td>
</tr>
<tr>
<td>measuring:</td>
<td>- Cluster structures</td>
</tr>
<tr>
<td>- Dyadic TM</td>
<td>- Brokers, core members and other</td>
</tr>
<tr>
<td>- Individual TM</td>
<td>important network members using</td>
</tr>
<tr>
<td></td>
<td>the centrality measures</td>
</tr>
</tbody>
</table>

memory systems, than an offshore outsource strategy. Software transfers, delegation of responsibilities and task assignments affect the extent people are aware of each others’ expertise, access the right people when needed, and trust the information they receive. As a result, certain members are better connected with others and they play a central role within the network.

Figure 10.1 presents our solution towards answering the main research questions. We conclude that the methods presented in this book, can be used as a “cause-and-effect” tool (such as the one presented in chapter 7) in order to analyze GSD governance decisions and their effect in transactive memory systems.

10.2 Current contributions and Future Directions

“Software is created by people for people working in a range of environments and under various conditions” [156]. Understanding therefore the human aspects of software engineering is imperative and this thesis builds upon that knowledge. Herbsleb [77] argued that computer science is necessary but not sufficient to understand and overcome contemporary problems in software engineering. We need to understand the human capabilities and enhance them in the context of software engineering; for this reason, Herbsleb [77] proposed to start creating a culture of interdisciplinary research. Following this advice, we performed an interdisciplinary research and proposed the use of theories from the social and business sciences in order to understand phenomena in the field of software engineering and particularly, in global software development. In the following
10.2. CURRENT CONTRIBUTIONS AND FUTURE DIRECTIONS

paragraphs, we discuss the contributions of this thesis from both a practical and theoretical perspective.

10.2.1 Contextualizing awareness in GSD

The first contribution of this thesis is the use of transactive memory systems as a way to measure expertise and team awareness in GSD. As we discussed in the previous chapters, awareness is necessary to coordinate group activities and ensure that individual contributions are relevant to the whole group. At the same time, awareness is a cognitive concept and as such is not easy to capture and measure. Originating from the field of psychology, we find that TMS theory serves our purpose, providing a systematic way to identify and measure team awareness. Additionally, TMS theory focuses on the specific (expertise) knowledge that each member possesses rather than on the common knowledge that all members share. Hence, from a practical point of view using transactive memory systems can improve GSD collaboration performance and productivity. As long as software is being built by people, these people need to know each other, to access and trust each other and of course, to communicate with each other. All these transactions can be identified and measured using our proposed method. Transactive memory systems can facilitate team building, enforce relationships between people from different geographic locations and improve coordination.

From a theoretical perspective, we also add to the body of knowledge of TMS by proposing a novel, twofold approach (a structural and a process approach) to measure transactive memory systems. Our contribution is interesting from a theoretical point of view, because we focus not only on the processes (transactions) that exist within TMS but also on the structure of those systems. We combined the results from both approaches into a common, whole definition of transactive memory systems. We showed that it is not only important to analyze how people retrieve or allocate expertise knowledge but also who are those people that are more likely to do so. We therefore argue that in order to understand better the development of transactive memory systems, one should consider both “sides of the coin”. Although this thesis focuses on global software development, we recognize that communication, collaboration and knowledge management challenges exist in any environment where people work together to accomplish knowledge-intensive tasks. Despite the fact that our proposed method was developed and applied in the context of GSD, we argue that it is not restricted to the field of distributed collaboration, neither to the area of software engineering. Hence, we also encourage the adoption and implementation of our methods in different business and social contexts.
10.2.2 “Drawing” the GSD collaboration

Our second contribution lies on the use of social network analysis methods to explore collaboration in global software development. Networks are a general yet powerful means of representing patterns of connections or interactions [136]. In this thesis, we used SNA methods to explore the interactions between distributed teams, how remote members are connected with each other and what kind of collaboration patterns exist. The use of social network theory in GSD proved beneficial in several ways. First, SNA methods allowed us to “draw” and visualize the collaboration between distributed teams. These visualizations give a common ground for discussion with people from all levels in the organization. For example, our empirical results suggest that high level managers find the visualizations useful because they can have a quick, holistic overview of their GSD teams. Software developers, testers, architects and team leaders find the visualizations useful because they can identify whether the current representation of their way of working is a desired one. Second, SNA methods helped us identify the important members of the GSD network. Identifying the important members is beneficial because practitioners may question, for instance, whether they want the software architect to become a broker between locations because he has a comprehensive knowledge of the software that’s being built, or the team leader because he has the “management” responsibility to take decisions and coordinate activities. A third practical contribution of SNA is the focus on the dynamic nature of global software development. As we discussed in the previous paragraphs, building software is all about people and teams, where memberships are changing, relationships evolve and consequently global software development becomes a volatile activity. Using SNA, we were able to capture “snapshots” of the collaboration activities over time, and observe their evolution. This observation may be for instance useful for the offset of GSD collaborations; one can capture a first “snapshot” of the network at the start of the collaboration and several others in different points in time in order to compare them, evaluate them and potentially decide upon their future development.

The use of social network theory in GSD proved also beneficial from a theoretical perspective. First, although the use of social network analysis in software engineering is not new, it is still not popular especially in industrial case studies. The main reason is that collecting SNA data is a time-consuming process and involves the commitment of the participants. Hence, this thesis contributes to the current research in GSD by providing empirical results of the SNA application in a set of industrial case studies. Second, analyzing GSD collaboration in different settings revealed an interesting contribution to GSD theory; the research-focused projects as a different type of projects in distributed software development. The results from our social network analysis revealed that research projects exhibit
10.2. CURRENT CONTRIBUTIONS AND FUTURE DIRECTIONS

characteristics from both GSD commercial and OSS projects. Although, our ob-
servations are limited to the SNA insights, we argue that these type of projects
can be considered a third type of distributed software development projects and
as such they should be further explored. Third, SNA gave us the opportunity
to examine GSD collaborations not just based on the individual characteristics
(i.e. characteristics of the network members) but also based on the dyadic rela-
tionships (i.e. the link between two network members). This is a novel approach
compared to the traditional research methods where only individual data are col-
lected. We therefore add to the current literature of GSD with observations and
results on distributed collaborations focusing not only on people but also on the
relationships between them. Finally, we applied a variety of SNA techniques in
our case studies but social network theory offers a plethora of other methods,
which might offer additional, interesting insights in GSD collaboration structures.
For instance, considering the directionality of the relationships measured in social
networks one can use the method proposed by [66], to identify different types of
brokers such as coordinators, liaisons, consultants and gatekeepers. We there-
fore encourage future studies to enhance the present thesis and explore more
possibilities on the SNA applications.

10.2.3 Managing GSD

Finally, we further contributed to the field of GSD with a simple and yet useful
governance model. Essentially, we suggest managers to consider three questions
when analyzing their GSD activities; (1) what is (or will be) the business strategy
between remote offices? (2) how tasks are (or will be) allocated? and (3) how
teams are (or will be) structured and organized? We used these three simple
questions in several case studies and our empirical observations suggest that they
can provide a clear insight on the basic GSD activities of the particular cases,
but also an effective mechanism to compare cases. After all, “governance is about
considering the efficacy of alternative modes (means) of organizing a business and
selecting the best mechanisms to suit the circumstances at hand” [5]. Combining
these three simple aspects of governance with the aforementioned concepts on
TMS and SNA, scholars and practitioners can use our proposed methods as a
“cause and effect” tool (such as the one proposed in chapter 7) to evaluate GSD
collaborations. From a theoretical perspective, one can perform further studies
where more governance modes are applied and potentially have different effects
on TMS. Eventually, a knowledge library can be created with different “cause and
effect” cases that can be used as a reference to other studies. From a more prac-
tical point of view, GSD practitioners can use the knowledge library to compare
and evaluate their current situation and assess future transformations.
CHAPTER 10. CONCLUSION

10.3 Limitations

The credibility of a study refers to the degree of confidence we have on its conclusions [150]. Thus, it is important to point out several aspects that might be a threat. We do so in order to establish the integrity and credibility of our observations and results.

To tackle internal validity in this thesis, we used several methods of data collection and analysis and we supported conclusions from one method, with results from the other. For instance, in chapters 5 and 7, we collected quantitative and we analyzed them using social network analysis as well as statistical analysis techniques. We further supported our observations with qualitative data from interviews, and focus meetings. Nevertheless, collecting data using questionnaires might be biased from the respondents individual perceptions of reality. We propose that a replication of the method used in this thesis by collecting data from other sources such as documentation or email communication will improve the internal validity. Furthermore, the absence of certain individual attributes of the participants, such as the years of experience working in the company/project, might also influence internal validity. For instance, someone who is working for many years in the project might have a better overview of the project, its members and the knowledge flow within the network, compared to newly introduced members. Cultural aspects might also influence people’s responses both during the interviews and when filling in the online survey. In this thesis, we do not take these aspects into account, but we suggest their consideration for the future.

Another threat to the reliability in research is the construct validity. Construct validity refers to the correct operational measures for the concepts being studies. This thesis makes two contributions; (1) a governance model in GSD and (2) a model to measure TMS in GSD. While both models have been applied to various case studies, we recognize the theoretical concepts used for the operationalization of the models be further re-evaluated, tested, validated and potentially improved.

Finally, we need to be cautious on how broadly we interpret and generalize the results. Although we used different case studies, our observations might be influenced by particular and distinctive characteristics of the companies, such as the company culture, trust issues between remote locations and others. Whereas “a theory may never be generalized to a setting where it has not yet been empirically tested and confirmed” [108], we suggest that using additional case studies to the research can eliminate the threat to external validity.
Concluding this four-year research program, we learn that “people are data too” [20]. Managing global software development activities is not just about processes and procedures but above all is about the people. Our results provide an insight to high- as well as middle-level managers on how their decisions affect software engineers, architects, testers, team leaders and all those who build software in distributed environments. Research-wise, we encourage future interdisciplinary studies where theories and concepts from one field can enforce results and observations in another.

Εν οίδα ότι ουδέν οίδα
(I know one thing, that I know nothing)
Socrates
Samenvatting

Het effect van Governance op Internationale Softwareontwikkeling: Een analyse van Transactive Memory Systems.

“Software wordt ontwikkeld voor mensen, door mensen in uiteenlopende omgevingen en omstandigheden” [156]. Het is daarom noodzakelijk om de menselijke kant van de softwareontwikkeling beter te begrijpen. Dit proefschrift draagt daaraan bij. We hebben een multidisciplinair onderzoek gedaan met een focus op het menselijke aspect waarbij theorieën uit zowel de sociale als de economische wetenschappen zijn gecombineerd om een beter begrip te krijgen van softwareontwikkeling in het algemeen en internationale softwareontwikkeling (Global Software Development; GSD) in het bijzonder. In dit proefschrift staat beschreven hoe wij met behulp van socialnetwerkanalyse het effect van verschillende managementvormen in de softwareontwikkeling op Transactive Memory Systems (TMS) hebben onderzocht.

Dit onderzoek presenteert een eenvoudig en bruikbaar managementmodel en draagt daarmee bij aan de kennis rondom internationale softwareontwikkeling. Wij adviseren managers om, wanneer zij hun internationale softwareontwikkelactiviteiten analyseren, een antwoord te formuleren op de volgende drie vragen: (1) wat is (of wordt) de samenwerkingsvorm tussen de verschillende ontwikkellocaties? (2) Hoe worden de ontwikkeltaken verdeeld? (3) Hoe gaat de teamstructuur eruit zien? Met behulp van deze drie vragen hebben wij meerdere bedrijven en projecten onderzocht. De empirische resultaten lijken erop te duiden dat de antwoorden op deze vragen niet alleen een goed inzicht geven in de internationale softwareontwikkelstructuren van de onderzochte bedrijven, maar dat de antwoorden tevens gebruikt kunnen worden om verschillende managementvormen met elkaar te vergelijken. Dit is waar governance in de basis om draait. Anders gezegd: “het onderzoeken van de doeltreffendheid van verschillende management-

In ons onderzoek gebruiken we Transactive Memory Systems (TMS) als een manier om de expertise en het team-bewustzijn in de internationale softwareontwikkeling te meten. Team-bewustzijn is noodzakelijk om groepsactiviteiten te coördineren en om zeker te stellen dat de resultaten van individuele teamleden bijdragen aan het doel van de groep. Het meten van team-bewustzijn is niet eenvoudig, met name omdat het een zeer zacht, psychologisch concept is. Echter, met behulp van de TMS-theorie, zoals deze is ontwikkeld in de psychologie, hebben we een manier gevonden om team-bewustzijn systematisch te analyseren en te meten. Naast het meten van team-bewustzijn biedt TMS-theorie ook de mogelijkheid om het kennisniveau van de individuen in een groep te bepalen. Door TMS in te zetten in onderzoek naar GSD is het mogelijk om de samenwerking te verbeteren en de productiviteit te verhogen. Zolang softwareontwikkeling mensenwerk is, is het belangrijk dat de mensen elkaar kennen, elkaar weten te vinden, elkaar vertrouwen en met elkaar blijven communiceren. Deze verschillende “transacties” kunnen worden gemeten en geanalyseerd met behulp van de technieken die gepresenteerd worden in dit proefschrift. TMS kan daarmee gebruikt worden om een sterkere teamband te bewerkstelligen, om relaties te versterken tussen collega’s die op verschillende locaties werken en om de algehele coördinatie te verbeteren.

Naast TMS gebruiken we ook socialenetwerkanalyse (SNA) om de samenwerkingen in GSD te verkennen. Netwerkanalyse is een algemeen maar krachtig middel om patronen van connecties en interacties te beschrijven [136]. In dit proefschrift gebruiken we SNA-methodes voor de analyse van de interacties tussen gedistribueerde teams, het onderzoeken van connecties tussen teamleden. Teven bekijken we verschillende samenwerkingsvormen van entwikkellokacities. Het gebruik van socialenetwerktheorie in GSD is in meerdere opzichten nuttig gebleken. SNA-technieken maakten het mogelijk om samenwerkingsvormen tussen gedistribueerde teams te visualiseren. Met deze visualisaties kunnen we samenwerkingsvormen bespreken met belanghebbenden in alle lagen van een organisatie. Verder biedt SNA ons de mogelijkheid om sleutelpersonen in GSD op te sporen. Een derde bijdrage van SNA aan ons onderzoek is dat het de analyse van het dynamische aspect van GSD. Zoals eerder aangegeven is softwareontwikke-
ling mensenwerk. Teams veranderen en relaties groeien. Als gevolg daarvan is GSD een zeer dynamische werkvorm. Met behulp van SNA is het mogelijk om “foto’s” te maken van samenwerkingen en deze te vergelijken met eerdere of latere momenten.

Gedurende dit vierjarige onderzoek zijn we tot de conclusie gekomen dat “mensen ook data zijn” [20]. Het managen van internationale softwareontwikkeling gaat niet alleen om processen en procedures maar vooral om mensen. Onze resultaten geven inzage in de beslissingen van top- en middelmanagement en in welke invloed deze beslissingen hebben op ontwikkelaars, architecten, testers, teamleiders en alle anderen die betrokken zijn bij het ontwikkelproces. Op het gebied van de wetenschap moedigen wij interdisciplinair onderzoek aan waarbij theorieën en concepten uit één veld leiden tot observaties en resultaten in andere velden.

English Summary

“Software is created by people for people working in a range of environments and under various conditions” [156]. Understanding therefore the human aspects of software engineering is imperative and this thesis builds upon that knowledge. Focusing on people, we performe an interdisciplinary research and propose the use of theories from the social and business science in order to understand phenomena in the field of software engineering and particularly, in global software development (GSD). This thesis focuses on the effect of global software governance on transactive memory systems (TMS) by taking a social network perspective.

We contribute to the field of GSD with a simple and yet useful governance model. Essentially, we suggest managers to consider three questions when analyzing their GSD activities; (1) what is (or will be) the business strategy between remote offices? (2) how tasks are (or will be) allocated? and (3) how teams are (or will be) structured and organized? We use these three simple questions in several case studies and our empirical observations suggest that they can provide a clear insight on the basic GSD activities of the particular cases, but also an effective mechanism to compare cases. After all, “governance is about considering the efficacy of alternative modes (means) of organizing a business and selecting the best mechanisms to suit the circumstances at hand” [5]. Combining these three simple aspects of governance, scholars and practitioners can use our proposed methods as a “cause and effect” tool to evaluate GSD collaborations. Eventually, a knowledge library can be created with different “cases” that can be used as a reference to other studies. From a more practical point of view, GSD practitioners can use the knowledge library to compare and evaluate their current situation and assess future transformations.
We also use transactive memory systems as a way to measure expertise and team awareness in GSD. Awareness is necessary to coordinate group activities and ensure that individual contributions are relevant to the whole group. At the same time, awareness is a cognitive concept and as such is not easy to capture and measure. Originating from the field of psychology, we find that TMS theory serves our purpose, providing a systematic way to identify and measure team awareness. Additionally, TMS theory focuses on the specific (expertise) knowledge that each member possesses rather than on the common knowledge that all members share. Hence, using transactive memory systems can improve GSD collaboration performance and productivity. As long as software is being built by people, these people need to know each other, to access and trust each other and of course, to communicate with each other. All these transactions can be identified and measured using our proposed method. Transactive memory systems can facilitate team building, enforce relationships between people from different geographic locations and improve coordination.

Furthermore, we use social network analysis (SNA) to explore collaboration in global software development. Networks are a general yet powerful means of representing patterns of connections or interactions. In this thesis, we use SNA methods to explore the interactions between distributed teams, how remote members are connected with each other and what kind of collaboration patterns exist. The use of social network theory in GSD proved beneficial in several ways. First, SNA methods allowed us to “draw” and visualize the collaboration between distributed teams. These visualizations give a common ground for discussion with people from all levels in the organization. Second, SNA methods help us identify the important members of the GSD network. A third contribution of SNA is the focus on the dynamic nature of global software development. Building software is all about people and teams, where memberships are changing, relationships evolve and consequently global software development becomes a volatile activity. Using SNA, we were able to capture “snapshots” of the collaboration activities over time, and observe their evolution.

Concluding this four-year research program, we learn that “people are data too”. Managing global software development activities is not just about processes and procedures but above all is about the people. Our results provide an insight to high- as well as middle-level managers on how their decisions affect software engineers, architects, testers, team leaders and all those who build software in distributed environments. Research-wise, we encourage future interdisciplinary studies where theories and concepts from one field can enforce results and observations in another.
Bibliografie


[38] Lee J Cronbach. Statistical tests for moderator variables: Flaws in analyses recently proposed. 1987. (Cited on page 120.)


[41] Kevin Crowston and James Howison. The social structure of free and


ICGSE ’06. International Conference on, pages 149 –158, 2006. (Cited on pages 60 and 63.)


[97] Barbara A Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. 2007. (Cited on pages 9 and 54.)


[156] Rafael Prikladnicki, Rashina Hoda, Marcelo Cataldo, Helen Sharp, Yvonne Dittrich, and Cleidson R. B. De Souza. 6th international workshop on


[176] Amrit Tiwana. An empirical study of the effect of knowledge integration on


[180] Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994. (Cited on pages [3], [52], [58], [71], [72], [93], and [99].)


[185] Claes Wohlin and Darja Smite. Classification of software transfers. In *Pro-


[189] Robert K. Yin. *Case Study Research: Design and Methods.* SAGE Publications, 2003. (Cited on pages [8], [10], [22], [39], [72], and [129])


<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>DEGAS - An Active, Temporal Database of Autonomous Objects</td>
<td>Johan van den Akker (CWI)</td>
</tr>
<tr>
<td></td>
<td>Information Retrieval by Graphically Browsing Meta-Information</td>
<td>Floris Wiesman (UM)</td>
</tr>
<tr>
<td></td>
<td>A Contribution to the Linguistic Analysis of Business Conversations</td>
<td>Ans Steuten (TUD)</td>
</tr>
<tr>
<td></td>
<td>within the Language/Action Perspective</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Memory versus Search in Games</td>
<td>Dennis Breuker (UM)</td>
</tr>
<tr>
<td></td>
<td>Computerondersteuning bij Straftoemeting</td>
<td>E.W.Oskamp (RUL)</td>
</tr>
<tr>
<td>1999</td>
<td>Physiology of Quality Change Modelling; Automated modelling of Quality</td>
<td>Mark Sloof (VU)</td>
</tr>
<tr>
<td></td>
<td>Change of Agricultural Products</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Classification using decision trees and neural nets</td>
<td>Rob Potharst (EUR)</td>
</tr>
<tr>
<td></td>
<td>The Nature of Minimax Search</td>
<td>Don Beal (UM)</td>
</tr>
<tr>
<td></td>
<td>The practical Art of Moving Physical Objects</td>
<td>Jacques Penders (UM)</td>
</tr>
<tr>
<td></td>
<td>Empowering Communities: A Method for the Legitimate User-Driven</td>
<td>Aldo de Moor (KUB)</td>
</tr>
<tr>
<td></td>
<td>Specification of Network Information Systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Re-design of compositional systems</td>
<td>Niek J.E. Wijngaards (VU)</td>
</tr>
<tr>
<td></td>
<td>Verification support for object database design</td>
<td>David Spelt (UT)</td>
</tr>
<tr>
<td></td>
<td>Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism</td>
<td>Jacques H.J. Lenting (UM)</td>
</tr>
<tr>
<td></td>
<td>for Discrete Reallocation</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>Perspectives on Improving Software Maintenance</td>
<td>Frank Niessink (VU)</td>
</tr>
<tr>
<td></td>
<td>Prototyping of CMS Storage Management</td>
<td>Koen Holtman (TUE)</td>
</tr>
<tr>
<td></td>
<td>Sociaal-organisatorische gevolgen van kennistechnologie; een</td>
<td>Carolien M.T. Metselaar (UVA)</td>
</tr>
<tr>
<td></td>
<td>procesbenadering en actorperspectief</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ETAG, A Formal Model of Competence Knowledge for User Interface Design</td>
<td>Geert de Haan (VU)</td>
</tr>
<tr>
<td></td>
<td>Knowledge-based Query Formulation in Information Retrieval.</td>
<td>Ruud van der Pol (UM)</td>
</tr>
<tr>
<td></td>
<td>Programming Languages for Agent Communication</td>
<td>Rogier van Eijk (UU)</td>
</tr>
<tr>
<td></td>
<td>Decision-theoretic Planning of Clinical Patient Management</td>
<td>Niels Peek (UU)</td>
</tr>
</tbody>
</table>
2000-8 Veerle Coup (EUR)  
Sensitivity Analysis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)  
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)  
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)  
Scalable Distributed Data Structures for Database Management

2001-1 Silja Renooij (UU)  
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)  
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)  
Learning as problem solving

2001-4 Evgueni Smirnov (UM)  
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)  
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)  
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)  
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)  
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 Pieter Jan ’t Hoen (RUL)  
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)  
Modeling and Simulating Work Practice  
BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA)  
Knowledge Management: The Role of Mental Models in Business Systems Design

2002-01 Nico Lassing (VU)  
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)  
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)  
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)  
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)  
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)  
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)  
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel (KUB)
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerd (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
2003-17 David Jansen (UT)  
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)  
Learning Search Decisions

2004-01 Virginia Dignum (UU)  
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)  
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)  
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)  
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)  
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)  
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)  
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek (UM)  
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)  
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)  
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)  
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)  
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)  
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)  
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)  
Multi-Relational Data Mining

2004-16 Federico Divina (VU)  
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)  
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)  
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)  
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)  
Learning from Design: facilitating multidisciplinary design teams

2005-01 Floor Verdenius (UVA)  
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM)  
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language
2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data
2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing
2005-06 Pieter Spronck (UM)
Adaptive Game AI
2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems
2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications
2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages
2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry
2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes
2005-16 Joris Graaumans (UU)
Usability of XML Query Languages
2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components
2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks
2005-19 Michel van Dartel (UM)
Situated Representation
2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives
2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006

2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting
2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations
2006-03 Noor Christoph (VUA)
The role of metacognitive skills in learning to solve problems
2006-04 Marta Sabou (VU)
Building Web Service Ontologies
2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines
2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering
2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web
2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor’s Opinion
2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems
2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types
2006-12 Bert Bongers (VU)
Interactivation - Towards an e-ology of people, our technological environment, and the arts
2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents
2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign
Towards a Theory of Requirements Change
2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain
2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks
2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device
2006-18 Valentin Zhizhkun (UVA)
Graph transformation for Natural Language Processing
2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach
2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining
2006-21 Bas van Gils (RUN)
Aptness on the Web
2006-22 Paul de Vrieze (RUN)
Fundamentals of Adaptive Personalisation
2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web
2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources
2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC
2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval
2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories
2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval

2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures
2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach
2007-03 Peter Mika (VU)
Social Networks and the Semantic Web
2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems
a dialogue-based approach
2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework
for Agent-enabled Surveillance
2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs
2007-07 Natasa Jovanovic (UT)
To Whom It May Concern - Addresssee Identification in Face-to-Face Meetings

2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations

2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology

2007-14 Niek Bergboer (UM)
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems

2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice

2007-18 Bart Orriens (UvT)
On the development and management of adaptive business collaborations

2007-19 David Levy (UM)
Intimate relationships with artificial partners

2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network

2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramírez Camps (CWI)
Structural Features in XML Retrieval

2007-25 Joost Schalken (VU)
Empirical Investigations in Software Process Improvement

====

2008
====

2008-01 Katalin Boer-SorhÁqan (EUR)
Agent-Based Simulation of Financial Markets: A modular, continuous-time approach

2008-02 Alexei Sharpanskykh (VU)
On Computer-Aided Methods for Modeling and Analysis of Organizations

2008-03 Vera Hollink (UVA)
Optimizing hierarchical menus: a usage-based approach

2008-04 Ander de Keijzer (UT)
Management of Uncertain Data - towards unattended integration

2008-05 Bela Mutschler (UT)
Modeling and simulating causal dependencies on process-aware information systems from a cost perspective

2008-06 Arjen Hommersom (RUN)
On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective

2008-07 Peter van Rosmalen (OU)
2008-35 Ben Torben Nielsen (UvT)
  Dendritic morphologies: function shapes structure

====

2009

====

2009-01 Rasa Jurgelenaite (RUN)
  Symmetric Causal Independence Models

2009-02 Willem Robert van Hage (VU)
  Evaluating Ontology-Alignment Techniques

2009-03 Hans Stol (UvT)
  A Framework for Evidence-based Policy Making Using IT

2009-04 Josephine Nabukunya (RUN)
  Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-05 Sietse Overbeek (RUN)
  Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality

2009-06 Muhammad Subianto (UU)
  Understanding Classification

2009-07 Ronald Poppe (UT)
  Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-08 Volker Nannen (VU)
  Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-09 Benjamin Kanagwa (RUN)
  Design, Discovery and Construction of Service-oriented Systems

2009-10 Jan Wielemaker (UVA)
  Logic programming for knowledge-intensive interactive applications

2009-11 Alexander Boer (UVA)
  Legal Theory, Sources of Law & the Semantic Web

2009-12 Peter Massuthé (TUE, Humboldt-Universitaet zu Berlin)
  Operating Guidelines for Services

2009-13 Steven de Jong (UM)
  Fairness in Multi-Agent Systems

2009-14 Maksym Korotkiy (VU)
  From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15 Rinke Hoekstra (UVA)
  Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16 Fritz Reul (UvT)
  New Architectures in Computer Chess

2009-17 Laurens van der Maaten (UvT)
  Feature Extraction from Visual Data

2009-18 Fabian Groffen (CWI)
  Armada, An Evolving Database System

2009-19 Valentin Robu (CWI)
  Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets

2009-20 Bob van der Vecht (UU)
  Adjustable Autonomy: Controlling Influences on Decision Making

2009-21 Stijn Vanderloooy (UM)
  Ranking and Reliable Classification

2009-22 Pavel Serdyukov (UT)
  Search For Expertise: Going beyond direct evidence

2009-23 Peter Hofgesang (VU)
  Modelling Web Usage in a Changing Environment

2009-24 Annerieke Heuvelink (VUA)
  Cognitive Models for Training Simulations

2009-25 Alex van Ballegooij (CWI)
  "RAM: Array Database Management through Relational Mapping"

2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services
2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web
2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models
2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications
2009-30 Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage
2009-31 Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text
2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors
2009-33 Khiet Truong (UT)
How Does Real Affect Affect Affect Recognition In Speech?
2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method Engineering Approach
2009-35 Wouter Koolewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling
2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks
2009-37 Hendrik Drachslor (OUN)
Navigation Support for Learners in Informal Learning Networks
2009-38 Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context
2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets
2009-40 Stephan Raaajmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language
2009-41 Igor Berezheny (UvT)
Digital Analysis of Paintings
2009-42 Toine Bogers
Recommender Systems for Social Bookmarking
2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients
2009-44 Roberto Santana-Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations
2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful
2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

==== 2010 ====

2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter
2010-02 Ingo Wassink (UT)
Work flows in Life Science
2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents
2010-04 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments
2010-05 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems
2010-06 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI
2010-07 Wim Fikkert (UT)
Gesture interaction at a Distance
2010-08 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments

2010-09 Hugo Kielman (UL)
A Politiegegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-O rganizational Models

2010-16 Sicco Verweer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19 Henriette Cramer (UvA)
People’s Responses to Autonomous and Adaptive Systems

2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of data degradation

2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data

2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions

2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies

2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27 Marten Voulon (UL)
Automatisch contracteren

2010-28 Arne Koopman (UU)
Characteristic Relational Patterns

2010-29 Stratos Ideos(CWI)
Database Cracking; Towards Auto-tuning Database Kernels

2010-30 Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31 Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web

2010-32 Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems

2010-33 Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34 Teduh Dirghayyu (UT)
Interaction Design in Service Compositions

2010-35 Dolf Trieschnigg (UT)
Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36 Jose Janssen (OU)
Paving the Way for Lifelong Learning: Facilitating competence development through
a learning path specification
2010-37 Niels Lohmann (TUE)
Correctness of services and their composition
2010-38 Dirk Fahland (TUE)
From Scenarios to components
2010-39 Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents
2010-40 Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web
2010-41 Guillaume Chaslot (UM)
Monte-Carlo Tree Search
2010-42 Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach
2010-43 Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies
2010-44 Pieter Bellekens (TUE)
An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain
2010-45 Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services
2010-46 Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment
2010-47 Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples
2010-48 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets
2010-49 Jahn-Takeshi Saito (UM)
Solving difficult game positions
2010-50 Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives
2010-51 Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources
2010-52 Peter-Paul van Maanen (VU)
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention
2010-53 Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access

====

2011
====

2011-01 Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models
2011-02 Nick Tinnemeier(UU)
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language
2011-03 Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems
2011-04 Hado van Hasselt (UU)
Insights in Reinforcement Learning: Formal analysis and empirical evaluation of temporal-difference learning algorithms
2011-05 Base van der Raadt (VU)
Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.
2011-06 Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage
2011-07 Yujia Cao (UT)
Multimodal Information Presentation for High Load Human Computer Interaction
2011-08 Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues
2011-09 Tim de Jong (OU)

Contextualised Mobile Media for Learning
2011-10 Bart Bogaert (UvT)

Cloud Content Contention
2011-11 Dhaval Vyas (UT)

Designing for Awareness: An Experience-focused HCI Perspective
2011-12 Carmen Bratosin (TUE)

Grid Architecture for Distributed Process Mining
2011-13 Xiaoyu Mao (UvT)

Airport under Control. Multiagent Scheduling for Airport Ground Handling
2011-14 Milan Lovric (EUR)

Behavioral Finance and Agent-Based Artificial Markets
2011-15 Marijn Koolen (UvA)

The Meaning of Structure: the Value of Link Evidence for Information Retrieval
2011-16 Maarten Schadd (UM)

Selective Search in Games of Different Complexity
2011-17 Jiayin He (UVA)

Exploring Topic Structure: Coherence, Diversity and Relatedness
2011-18 Mark Ponsen (UM)

Strategic Decision-Making in complex games
2011-19 Ellen Rusman (OU)

The Mind ’ s Eye on Personal Profiles
2011-20 Qing Gu (VU)

Guiding service-oriented software engineering - A view-based approach
2011-21 Linda Terlouw (TUD)

Modularization and Specification of Service-Oriented Systems
2011-22 Junte Zhang (UVA)

System Evaluation of Archival Description and Access
2011-23 Wouter Wierkamp (UVA)

Finding People and their Utterances in Social Media
2011-24 Herwin van Velbergen (UT)

Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior
2011-25 Syed Waqar ul Quanain Jaffry (VU)

Analysis and Validation of Models for Trust Dynamics
2011-26 Matthijs Aart Pontier (VU)

Virtual Agents for Human Communication - Emotion Regulation and Involvement, Distance Trade-Offs in Embodied Conversational Agents and Robots
2011-27 Aniel Bhulai (VU)

Dynamic website optimization through autonomous management of design patterns
2011-28 Rinus Kaptein(UVA)

Effective Focused Retrieval by Exploiting Query Context and Document Structure
2011-29 Faisal Kamiran (TUE)

Discrimination-aware Classification
2011-30 Egon van den Broek (UT)

Affective Signal Processing (ASP): Unraveling the mystery of emotions
2011-31 Ludo Waltman (EUR)

Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
2011-32 Nees-Jan van Eck (EUR)

Methodological Advances in Bibliometric Mapping of Science
2011-33 Tom van der Weide (UU)

Arguing to Motivate Decisions
2011-34 Paolo Turrini (UU)

Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
2011-35 Maaike Harbers (UU)

Explaining Agent Behavior in Virtual Training
2011-36 Erik van der Spek (UU)

Experiments in serious game design: a cognitive approach
2011-37 Adriana Burlutiu (RUN)

Machine Learning for Pairwise Data, Applications for Preference Learning and
Supervised Network Inference
2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization
2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games
2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development
2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control
2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution
2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge
2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces
2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection
2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
2011-47 Azizi Bin Ab Aziz(VU)
Exploring Computational Models for Intelligent Support of Persons with Depression
2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2011-01 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda
2011-02 Muhammad Umair(VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
2011-03 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories
2011-04 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications
2011-05 Marijn Plomp (UU)
Maturing Interorganisational Information Systems
2011-06 Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks
2011-07 Rianne van Lambalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
2011-08 Gerben de Vries (UVA)
Kernel Methods for Vessel Trajectories
2011-09 Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms
2011-10 David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment
2011-11 J.C.B. Rantham Prabhakara (TUE)
Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
2011-12 Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems
2011-13 Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
2011-14
Evgeny Knutov(TUE)
Generic Adaptation Framework for Unifying Adaptive Web-based Systems
2012-15 Natalie van der Wal (VU)
Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes
2012-16 Fiemke Both (VU)
Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
2012-17 Amal Elgammal (UvT)
Towards a Comprehensive Framework for Business Process Compliance
2012-18 Eltjo Poort (VU)
Improving Solution Architecting Practices
2012-19 Helen Schonenberg (TUE)
What’s Next? Operational Support for Business Process Execution
2012-20 Ali Bahramisharif (RUN)
Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
2012-21 Roberto Cornacchia (TUD)
Querying Sparse Matrices for Information Retrieval
2012-22 Thijs Vis (UvT)
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
2012-23 Christian Muehl (UT)
Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction