Benefits of transactive memory systems in large-scale development

Aivars Sablis
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Aivars Sablis
E-mail: aivars.sablis@people.lv

University advisor:
Dr. Darja Šmite, Associate Professor
DIPT

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Internet : www.bth.se
Phone : +46 455 38 50
Fax : +46 455 38 50
00
57
ABSTRACT

Context. Large-scale software development projects are those consisting of a large number of teams, maybe even spread across multiple locations, and working on large and complex software tasks. That means that neither a team member individually nor an entire team holds all the knowledge about the software being developed and teams have to communicate and coordinate their knowledge. Therefore, teams and team members in large-scale software development projects must acquire and manage expertise as one of the critical resources for high-quality work.

Objectives. We aim at understanding whether software teams in different contexts develop transactive memory systems (TMS) and whether well-developed TMS leads to performance benefits as suggested by research conducted in other knowledge-intensive disciplines. Because multiple factors may influence the development of TMS, based on related TMS literature we also suggest to focus on task allocation strategies, task characteristics and management decisions regarding the project structure, team structure and team composition.

Methods. We use the data from two large-scale distributed development companies and 9 teams, including quantitative data collected through a survey and qualitative data from interviews to measure transactive memory systems and their role in determining team performance. We measure teams’ TMS with a latent variable model. Finally, we use focus group interviews to analyze different organizational practices with respect to team management, as a set of decisions based on two aspects: team structure and composition, and task allocation.

Results. Data from two companies and 9 teams are analyzed and the positive influence of well-developed TMS on team performance is found. We found that in large-scale software development, teams need not only well-developed team’s internal TMS, but also have well-developed and effective team’s external TMS. Furthermore, we identified practices that help of hinder development of TMS in large-scale projects.

Conclusions. Our findings suggest that teams working in large-scale software development can achieve performance benefits if transactive memory practices within the team are supported with networking practices in the organization.

Keywords: Large-scale; software development; transactive memory system; empirical study; multi-case study; knowledge management; TMS; team performance; distributed; global software engineering; expertise coordination;
CONTENTS

ABSTRACT .......................................................................................................................... 1

CONTENTS ......................................................................................................................... II

1 INTRODUCTION ............................................................................................................. 1
  1.1 PROBLEM OUTLINE ................................................................................................. 1
  1.2 BACKGROUND AND MOTIVATION ......................................................................... 2
  1.3 RESEARCH AIMS, OBJECTIVES AND RESEARCH QUESTIONS .............................. 3
  1.4 RELEVANCE ........................................................................................................... 4
  1.5 THESIS STRUCTURE ............................................................................................... 4

2 RELATED WORK ............................................................................................................. 6
  2.1 RELATED WORK IN KNOWLEDGE MANAGEMENT ............................................. 6
  2.2 RELATED WORK IN COGNITIVE SCIENCES AND TRANSACTIVE MEMORY SYSTEMS ... 8
  2.3 HOW TO MEASURE TRANSACTIVE MEMORY SYSTEMS .................................. 10
  2.4 RESEARCH GAPS ................................................................................................ 11

3 RESEARCH APPROACH ............................................................................................... 13
  3.1 RESEARCH METHODOLOGY ................................................................................ 13
  3.2 SURVEY ................................................................................................................ 15
    3.2.1 Survey questions ............................................................................................... 15
    3.2.2 Survey sampling .............................................................................................. 16
    3.2.3 Survey execution .............................................................................................. 16
    3.2.4 Response rate .................................................................................................. 16
    3.2.5 Avoiding Bias .................................................................................................. 17
    3.2.6 Survey data preparation .................................................................................. 17
  3.3 DATA ANALYSIS .................................................................................................... 18
    3.3.1 Survey data analysis ....................................................................................... 18
    3.3.2 Team performance data analysis .................................................................... 19
    3.3.3 Correlation and regression analysis ................................................................. 20
    3.3.4 Focus group data analysis ............................................................................... 22
  3.4 RESULTS .................................................................................................................. 22
  3.5 DISCUSSION ........................................................................................................... 24
    3.5.1 TMS in large-scale software development projects ........................................ 24
    3.5.2 Implications for Practice ................................................................................. 24
    3.5.3 Validity Threats .............................................................................................. 25

4 CONCLUSION AND FUTURE WORK ........................................................................... 27

5 ACKNOWLEDGMENT .................................................................................................. 29

6 REFERENCES ................................................................................................................. 30

7 APPENDIX I – PAPER .................................................................................................. 33

8 APPENDIX II - QUESTIONNAIRE ................................................................................. 56

9 APPENDIX III – SURVEY INSTRUCTIONS .................................................................. 57

10 APPENDIX IV – FOCUS GROUP QUESTIONS ............................................................. 58

11 APPENDIX V – TEAM PERFORMANCE EVOLUTION QUESTIONS ............................ 61

12 APPENDIX VI – SYSTEMATIC LITERATURE REVIEW RESULTS ............................ 62
1 INTRODUCTION

1.1 Problem Outline

Today, when large-scale software developments have become more common, as software grows in size and complexity, the question of how to cope with the challenges to knowledge coordination in managing those projects and still achieving business needs is very important. Companies that do large-scale development often need to overcome the challenges related to globally distributed development as well, such as coordination and accessing knowledge between sites, when having fewer formal and informal interactions between team members [1]. They have to address challenges which occur due to poor communication channels and psychological interactions and can reduce their awareness of others’ work tasks [2]. Large-scale software development projects are those consisting of a large number of teams (in [54] suggested more than 2), maybe even spread across multiple locations, and working on large and complex software development. Challenges in such projects are that neither a team nor any member in the project can possibly possess all the knowledge required for successful work. As software engineering is knowledge-intensive work [4], challenges to communication and coordination and finding expertise are even bigger issues in large software engineering projects [5]. Companies choose to establish their own offshore insourcing sites and large projects are more than often developed by teams that are distributed globally. However, these companies still have to overcome a lot of challenges related to globally distributed development teams and many companies are still trying to increase performance of large-scale distributed software development projects [1][6].

These challenges are tied to that large-scale software development human aspects, such as communication in a team and coordination between teams and across sites are more important than the technological aspects for better performance [7]. Theories of cognitive science could be particularly relevant in this respect. It is argued that organizational learning is a source of competitive advantage. It enables companies with ability to continuously adapting and integrating its key assets and competences in new environments and researchers have called for more work in this field [8]. At the same time knowledge management has become more actual topic in software development [7]. However, only a few studies have focused on knowledge management in large-scale or distributed software organizations [4]. These studies focus on the role of networks to improve ways of working [55] and software processes [56], and on personal networks [57]. However, we did not find papers focusing on product knowledge sharing in team or organizational networks.

The core focus of this thesis is the application of transactive memory system theory [9], that can be described as “who knows what”. For our study the dimensions covered in the theory, i.e. knowledge specialization, knowledge coordination and credibility [10], are really relevant in addressing the challenges of large-scale software development and could provide us with a better understanding about social and knowledge network characteristics in large-scale projects. Also, previous studies have shown the positive relationship between transactive memory systems and group performance [11]. If relevant in such projects, transactive memory system could provide theoretical foundations on how we can manage knowledge within large-scale project networks and what type of actions we should take when organizing teams and projects to get most benefits. However, while there has been a reasonable body of studies about transactive memory theory in different contexts, we identified only a few studies [11][43] in the context of software development. Thus, we seek to address this research gap.

To answer our research question (presented in Section 1.3) we have conducted a social network analysis survey to measure transactive memory system with latent variable model and capture structural relations of the team members inside and outside of the team, for a number of teams.
in two companies, and aimed to understand why these relations occur and what are their consequences.

### 1.2 Background and Motivation

This research is a part of the TEDD\(^1\) project, that aims at understanding how to improve distributed development with the focus on efficiency and quality. Through the TEDD project the author of this thesis was able to access the data from two large-scale distributed projects from two companies.

**Company A** develops generic software products offered to an open market and complex compound systems with customised versions. Company A has worked in agile development almost 10 years and today agile practices have become the way of working. The studied project in Company A is a sub-system that has multiple components and interfaces with other sub-systems. Sub-system is very complex and contains several millions lines of code. 8 years ago they realized that they lack a competence in certain areas and experienced problems implementing new features in the large-scale system. The size and complexity of the system domain and product knowledge requires years to become knowledgeable developer. This facilitated shift towards cross-functional development teams that are responsible for a feature from high level description of a feature until its release to customer. Development work is organized in seventeen self-managing cross-functional feature teams that comprise of members with different roles. The number of developers working on the sub-system grew from eight developers in 2007 to up to around 60 developers by 2013. In early 2014, there were five teams in Sweden, eight in-house teams and two homeshore consultant teams in China, and two Korean teams. To manage releases, program management, product owners, and release management support the teams. Through TEDD project author was able to gain data from 4 teams in Sweden and 3 teams in China.

**Company B** develops automated software solutions and embedded software for various industries. Company B puts large emphasis on product and process quality. The studied project in Company B is a complex system with multiple modules that have coupled dependencies and requires integration of knowledge about various areas and functionality. System contains several millions lines of code and is written in multiple programming languages. For safety critical requirement reasons, project follows a V-model development methodology were tasks are structured and project team roles are module-specific. Due to size and complexity of system, a good developer is said to have about a 50% overview of system. The main Swedish location employs the majority of software developers, and thus software projects are usually led from this location with sub-project managers in each location. Hardware expertise resides in the second Swedish location, and thus hardware projects are led from there. Development work is organized in projects, which can include development of new generations of the system, new functionality development, roll-ups of large maintenance projects and pure maintenance projects. In the early 2014, the work was distributed across two sites in Sweden and a site in India, all branches of Company B. Swedish site, referred to as Sweden 1 in the results, is responsible for the product. In general, there are six software teams in the main development site in Sweden, a software team in the secondary site in Sweden and a group of developers in India. Both Swedish sites were engaged in the system development from the very beginning, while the Indian site joined in 2006. Through TEDD project author was able to gain data from 3 teams in Sweden from both Swedish sites and a group of developers in India.

\(^1\) http://bth-tedd.weebly.com
When software projects have multiple teams they become large-scale and grow in size and complexity. When large-scale projects grow up to the size where nobody can know the whole system it creates challenges to project management with respect to communication, work coordination and knowledge needs of software development teams. This creates the need for knowledge management within teams and in the organization as a whole. Software organizations usually try to break the project into loosely coupled teams, but that creates new challenges when tasks and responsibilities span multiple teams or areas. Some software organizations try to address that by allocating dedicated experts in certain areas of system and creating knowledge coordination forums, such as communities of practices. However, all this increases the need of networking to gain access to valuable knowledge to solve tasks for individuals and teams. To facilitate efficiency of knowledge networks software companies, need to employ a variety of knowledge management practices. Knowledge management is “a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company’s knowledge” [21]. While in software development there is only a few studies regarding this topic, we can use theories and lessons learned from cognitive science field. The theory of transactive memory system (TMS) describes how group members allocate, store and retrieve knowledge within group. This theory is commonly referred as “group mind” [9]. Previous research of TMS shows positive benefits of well-developed TMS in groups for group performance, creativity and learning. However, as groups benefit somewhat different depending on tasks they are required to accomplish [45] we need to examine whether large-scale software development teams have positive benefits from well-developed TMS. If we can identify that large-scale software teams benefit in the same way as suggested in previous research, we can apply the lessons learned from other TMS research to facilitate the development of TMS in large-scale software organizations and achieve better team performance.

1.3 Research Aims, Objectives and Research Questions

The aim of this thesis is to understand to whether teams in large-scale development can benefit from well-developed transactive memory system in terms of team performance as is the case for teams in other disciplines. At the end, we are interested to understand the implications for team-building principles, task allocation and addressing knowledge needs of teams and how to create effective knowledge networks within organization.

Our proposed objectives are:
1. Understand the type of knowledge networks and identify social network characteristics in large-scale software development companies.
2. Identify whether well-developed TMSs in large-scale development leads to the benefits in performance, as promised by studies from other knowledge-intensive field settings.
3. Identify how different management decisions (such as team configuration, task allocation, support of team networking practices) affect the structure of TMSs as well as the processes developed within those systems in large-scale software development organizations.

We seek to understand the potential of the organizational management practices from cognitive sciences help in improving performance in large-scale distributed software development. Thus, we seek to address following research questions:

RQ1: How does a well-development TMS influence team performance in large-scale development?

RQ2: What influences the development of a TMS in large-scale software development?
1.4 Relevance

What are we studying?
We are studying transactive memory systems for teams in large-scale distributed software development projects, and their links with perceived team performance. We conducted a survey and structured focus groups in order to answer our research questions. We studied the effect of well-developed transactive memory systems on perceived team performance and found that in large-scale distributed software development teams need not only well-developed transactive memory within team, but also within other teams and experts. We also found that number of unique contacts of that team members have influences performance.

Why are we interested in it?
The large scale software development teams often face problems with accessing knowledge in a timely manner and smooth execution of the tasks. In order to achieve the performance and to overcome the problems posed by the context of large-scale distributed software projects, effective expertise location techniques should be established within these companies. Knowledge network characteristics in software development have an impact on how manage knowledge within such network and what type of actions we should take to get most benefits. We want to understand the benefits of TMS in large-scale software development context, because it has been regarded as promising in other knowledge-intensive disciplines.

Why should this be interesting to others?
If we find that well-developed TMS of teams in large-scale software development leads to better performance of those teams, then it has implications for strategies of team formation, task allocation. For example, then practices how tacit information is stored as explicit information should be based on principles how group memories about project are stored and support transactive memory system network in organization. And if an organization is aware of its existing knowledge sharing network, it can make decision about how to use and develop further this network in most efficient way adjusting communication structures, meetings, communities of practices and allocating sufficient resources for such activities.

1.5 Thesis Structure

The remaining thesis is organized as follows.

Table I. Thesis structure

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Background</td>
<td>Provides the background study for the concepts used in this thesis.</td>
</tr>
<tr>
<td></td>
<td>Discussion of knowledge and business needs in software organizations,</td>
</tr>
<tr>
<td></td>
<td>shared mental models and transactive memory system and related concepts</td>
</tr>
<tr>
<td></td>
<td>in cognitive sciences. We also tried to link all these concepts</td>
</tr>
<tr>
<td></td>
<td>theoretically to give a clear picture of their relations.</td>
</tr>
<tr>
<td>3. Research Approach</td>
<td>In this chapter we provide the summary of the research approach</td>
</tr>
<tr>
<td></td>
<td>followed in this thesis, that is research methodology for the empirical</td>
</tr>
<tr>
<td></td>
<td>study presented in Paper-1</td>
</tr>
<tr>
<td>4. Summary of Research Results</td>
<td>We present our findings on the benefits of transactive memory</td>
</tr>
<tr>
<td></td>
<td>system in large-scale software projects. We demonstrate positive</td>
</tr>
<tr>
<td></td>
<td>effect of TMS on teams performance.</td>
</tr>
<tr>
<td>5. Conclusion and Future Work</td>
<td>In this chapter, we discuss the conclusions and future work of the</td>
</tr>
<tr>
<td></td>
<td>study.</td>
</tr>
<tr>
<td>Appendix I</td>
<td>The results of the study presented in this thesis are packed in a Paper-1, which can be found in Appendix I.</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Appendix II</td>
<td>Contains the survey questionnaire.</td>
</tr>
<tr>
<td>Appendix III</td>
<td>Contains survey instructions that was send to teams.</td>
</tr>
<tr>
<td>Appendix IV</td>
<td>Contains the list of interview questions.</td>
</tr>
<tr>
<td>Appendix V</td>
<td>Contains team evaluation questions.</td>
</tr>
<tr>
<td>Appendix VI</td>
<td>Results from Systematic Literature Review (SLR) of TMS benefits.</td>
</tr>
</tbody>
</table>
2 RELATED WORK

In this section we describe related research work. First, we describe how knowledge management in software engineering is related to business and knowledge needs in large-scale software organizations. Then we look for related work in knowledge management in general. As we see that software organizations could learn from knowledge management in cognitive sciences and search for relevant concepts and theories. Finally, our search is concentrated on transactive memory systems and to use transactive memory systems in our research we conducted an informal literature review both on transactive memory positive benefits and how to measure transactive memory in teams and organizations.

2.1 Related work in knowledge management

Before summarizing the related work in knowledge management, we first describe the business and knowledge needs of the software companies in the studied context. Large-scale distributed software projects are those consisting multiple teams in, possibly in multiple locations. Small size software applications can be simple and can be developed by a small number of people in the same location. However, if we increase the size and complexity of software systems, we are forced to scale up the number of teams that carries out all the development work. This creates challenges to project management and how to design teams, especially if software under development has multiple sub-systems or modules and can span many millions lines of code, as it is common is large embedded systems. Software projects face challenges due to the software product characteristics of scalability, uncertainty, interdependency and communication [1].

These challenges are augmented by additional challenges of large-scale development. Large-scale development include are communication, coordination of and access to knowledge across teams and experts when nobody can know everything and in case of distributed development - sites, when having fewer formal and informal interactions between team members. Communication and collaboration is related to mutual sharing of knowledge. Studies of teamwork show how shared knowledge improves team effectiveness, because sharing knowledge about the development process and what’s to be developed helps teams avoid costly misunderstandings. [6] This creates the need for knowledge management within teams and in the organization as a whole. Usually software organizations tend to break large project staff into small loosely coupled teams. These teams are designed to take responsibility of part of development process and/or system components. In relation to that group task requires coordination of their activities for successful completion of the task [12], additional difficulties of the large-scale distributed software development process by increasing the coordination need due to the distance [14]. As the distance between the teams grows they require more coordination effort [14] and thus there is the need to create new techniques to support coordination.

In the field of knowledge management in general the topic of management actions that support knowledge sharing, what some refer to as knowledge governance, is a much debated topic. How much should be formal, and what should be left to employees to organize themselves? Software organizations can view knowledge management as a risk prevention and mitigation strategy [20], because it can address risks that often are ignored in software development and especially in large-scale development. Examples of problems that knowledge management can address are:

- teams or team members create mistakes and need to perform rework because they didn’t have timely access to knowledge or didn’t know where to look for such knowledge, i.e. lack of awareness of who knows what.
- individuals and experts who own key knowledge become inaccessible or overloaded with requests to provide their knowledge to teams or team members,
long time of acquiring knowledge and sometimes inability to acquire all knowledge about the system, especially in large-scale development,

- loss of knowledge due to attrition.

Software companies have needs to stay on the market. These needs can be viewed as a business need and from a knowledge needs perspectives, as business needs create challenges that can be addressed by knowledge management. They increasingly strive to create better quality software in shorter time and keep project costs in check. Repeating successful processes would increase productivity, quality and the likelihood of further success. This requires software organizations to create and use an enormous amount of knowledge to support business objectives with technology, processes, projects, products and domain knowledge in software development areas [20]. In order to avoid repeating mistakes but to actively repeat successes, knowledge gained overtime could be used to guide and improve future tasks [15]. In reality development teams do not take full advantage of existing experience, but repeat mistakes over and over again [17]. Valuable individual experience is acquired with each project, and much more could be gained if there were a systematic way to efficiently share this diverse knowledge [16].

Software development process involves intensive decision making process [18] [19], where every person has to constantly make decisions of either technical or managerial nature. When a project is small, it is possible to use informally shared personal knowledge and experience. However, this is not feasible in large-scale and distributed projects involving many times more people and handling information over distance and possibly, across multiple time zones [19]. Thus, such individual knowledge for decision making should be shared and leveraged at the project and organization levels in organized fashion, and there is a wide belief that coordinating and complimenting efforts of formal ways of sharing knowledge with deliberate ways of sharing tacit knowledge are required.

Knowledge management is a discipline that promises to gain advantage from exploiting organization’s intellectual capital, knowledge that is embed in their human resources, networking and documentation and ways of working. Commonly used definition of knowledge management is “a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company’s knowledge” [21]. However, knowledge is a large interdisciplinary field and as a consequence, there is an ongoing debate as to what constitutes knowledge management [4]. A related term is organizational learning and it differs from individual learning in two respects [22]: first, it occurs through a shared insight, knowledge and shared models; second, it is based not only on the memory of the participants in the organization, but also on “institutional mechanisms” such as policies, strategies, explicit models and defined processes (we can call this the “culture” of the organization). These mechanisms may change over time, what we can say is a form of learning [4].

Software organizations need to capture knowledge in order to locate sources of knowledge. Some of the organizational knowledge is captured on different media (paper, electronic files, and so on) and this is explicit knowledge. However, far from all knowledge in an organization can be made explicit, and far from all explicit knowledge can be captured. Software organizations are heavily dependent on knowledge that lies within knowledgeable people [8]. Tacit knowledge is personal knowledge that employees gain through experience; this can be hard to express and is largely influenced by their beliefs, perspectives, and values [20]. Software organizations depend heavily on knowledgeable employees because they are key to the project’s success. These people are important for the success of projects, however it can be difficult to identify and access them. Software organizations need to develop knowledge maps and identify sources of knowledge in terms of know-who and know-where [58]. Teams must be able to ensure knowledge transactions effectively through expertise coordination, which requires knowing where expertise is located, knowing where expertise is needed, and integrating the right expertise wherever and whenever it’s needed [11]. Once such a knowledge map is in place, it can be used to identify appointed and de facto experts, staff new projects based on skills and experience required, and identify knowledge gaps that indicate the need to hire new people or to develop training programs [20]. A systematic review reported in [4] also found that communities built on existing networks are more likely to succeed [6]. These
knowledgeable people are also very mobile. When a person with critical knowledge suddenly leaves an organization, it creates severe knowledge gaps [20] — but probably no one in the organization is even aware of what knowledge they lost. Knowing what employees know is necessary for organizations to create a strategy for preventing valuable knowledge from disappearing.

2.2 Related work in cognitive sciences and transactive memory systems

Many organizations today rely on knowledge assets to differentiate their products and services. One way that organizations leverage these assets is with knowledge-worker teams, whose members use expertise and experience to solve problems and create intellective products [23]. Consulting teams, product development teams, research teams, and other cross-functional and ad hoc project teams are a few types of teams that are purposefully constructed to leverage the specialized expertise of individual team members [23]. Researchers have developed several theoretical frameworks of group knowledge processes in groups to help explain coordination and problem-solving dynamics within workgroups. Team mental models [24], information sampling models [25], team learning [26], and transactive memory systems [9] are some of the more active recent theoretical approaches to group knowledge. Previous studies have shown the positive relationship between transactive memory systems and group performance [23] [27], and has been also recently applied in software engineering, and therefore we focus on this theory in more detail in the following paragraphs.

Transactive memory could be described by "who knows what" [9] and is a combination of individual knowledge and interpersonal awareness and uniform agreement of others’ knowledge [28] - a mechanism through which groups collectively encode, store, and retrieve knowledge. Second part of transactive memory systems - the interpersonal awareness of others’ knowledge, have three basic processes of a transactive memory system: directory updating, information allocation and retrieval coordination [9]. Directory updating of 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, shown in Fig. 1. – upper branch. Information allocation in a transactive memory system entails the process of allocating knowledge to the person whose expertise will facilitate its storage, shown in Fig. 1. - middle branch. Additionally, in a knowledge network information is retrieved based on knowledge of the relative expertise of the individuals [29] – shown in Fig.1, lower branch. For overview of these processes see Fig I. Furthermore, success of directory updating process depends on team member ability to create memories about others in multiple ways. Team members can create memory entries based on their perception of other team members’ characteristics or perception of certain expertise areas of team members. Team members can also negotiate memory entries when planning or solving tasks together. Team members can create memory entries when they known that other team members have gained information beforehand or when it is known that team member may have been ability to gain certain expertise for longer time. These four types of directory updating is shown in Fig. 1.
Previous studies propose that transactive memory systems exist when three manifest variables are observed: specialization, coordination and credibility are aspects that can manifest the cognitive processes within a transactive memory system [30]. The reason that specialization, credibility, and coordination are observed together is because a transactive memory system is operating in the group. The manifest variables are independent after controlling for a transactive memory system (i.e. apart from the explanation that a transactive memory system is operating, there is no theoretical reason for the specialization, credibility, and coordination variables to be related) [23].

Specialization is explained as when members of a group have differentiated and specialized knowledge domains [9], e.g. the tendency for groups to delegate responsibility and to specialize in different aspects of the task [23] [31] [32]. Coordination is explained as the ability of team members to coordinate their work in effective, orchestrated way based on their knowledge of who knows what in the group [25] [31]. Credibility is explained as ability to rely on each other for their specialized knowledge and the tendency for groups to delegate responsibility and to specialize in different aspects of the task [28] [31]. Additionally, two other factors are found to be important to develop an effective transactive memory system. Expertise awareness - group members are aware of the expertise of each other and expertise accessibility - that 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 [33].

Our informal literature review found that previous studies on transactive memory systems shows that organisations and knowledge teams benefit in several areas from a well-developed transactive memory system. Early studies of transactive processes show that it helps members recall and discuss all available information and that it helps pool expertise that is distributed across members, which increases the chances that the group will find a solution or make a good decision with regard to a chosen task [34]. That has been observed in laboratory experiments with radio-kit assembly [27], [30], [35] and also in transactive memory systems in different field settings, such as top-management [36], and product development [23] [37]. Product teams that had higher transactive memory had positive effects in terms of higher quality products, products that better meet clients’ needs, and timely completion of client projects [23]. In product development teams it was demonstrated that a transactive memory system has a positive association with team learning, speed-to-market, and loyalty that exists between a provider and a consumer. When the project team established an effective TMS, it developed the new product with fewer technical problems, found and solved the product...
problem areas with which customers were dissatisfied, and developed and launched product faster and better. It was also partly demonstrated that the higher the task complexity, the more of an impact that TMS will have on team learning, speed-to-market, and new product success [37], sales [38] and product teams [28]. Product teams research demonstrated the group with higher transactive memory systems will be more successful in accomplishing its goals, will be evaluated better by external evaluators and that group will be evaluated better by its members [28]. The authors also demonstrated that the group with higher external relationship transactive memory system will be more successful in accomplishing its goals, will be evaluated better by external evaluators and also evaluated better by its members [28]. Large-scale and distributed knowledge-intensive work context partially is addressed in a sales company [33] and product development [39]. Knowledge intensive and stressful work is addressed in a study by air-traffic controllers [40]. Air-traffic controller groups research demonstrated that team member with greater experience working together will request and accept greater backup from one another and teammates who are in greater consensus regarding one another’s expertise will request and accept backup from one another to a greater extent than will those in lesser consensus [40] and also positive impact on team’s creativity [41]. These findings demonstrate that transactive memory systems fully mediate the effect of direct task experience on team creativity. Teams who acquired task experience directly are more creative because they develop better transactive memory systems than teams who acquired experience vicariously [41] and trust [42]. One exception was the specialization subscale, which appeared to behave differently in functional teams [23].

However, we found only two studies that were particularly addressing software engineering field, one with limitation to four cases and call for additional studies [11] and the other addressing globally distributed teams [43]. In a study of software development teams, it was found that expertise coordination within teams affected some performance measures but not others. The study finds that task transactive memory has a greater effect on goal attainment and external evaluation than on internal evaluation [11].

Generating group capabilities involves more than simply assembling a group of individuals with a wide range of specialized knowledge [42]. Although this joint knowledge pool may establish a strong foundation for a successful group, actual group performance depends upon how well the individual members are able to tap into the assembled knowledge pool and how well they are able to reconfigure this knowledge pool in new situations. Transactive memory, a group’s awareness of the location of knowledge resources distributed throughout the group, provides a promising approach for future study of knowledge and expertise in groups [28], and especially in groups working with high task complexity as demonstrated in [37]. These finding encourage future research of benefits of well-developed transactive memory systems in software engineering - how transactive memory system is impacted in software teams in large-scale projects.

### 2.3 How to measure transactive memory systems

Transactive memory system is a knowledge network that we can measure. To analyze a transactive memory in groups two types of measurements are used. Direct measurement is used to investigate transactive memory systems structure and indirect measurement is used to manifest transactive memory systems existence in teams. Latent variable model was developed by Lewis [23] and is used in several cases by other researchers [39] [41]. There are studies that use modified latent variable model [28] [41] as well as cases that used different independent variable models [33] [36] [43]. Latent variable model is based is based on hypothesized relationship between theoretic causes and effects of the test construct and on three variables, specialization, coordination and credibility. When members specialize in different domains of expertise, confidently rely on other members to accomplish joint tasks, and coordinate task processes, their team should utilize and integrate task-relevant knowledge more effectively [23]. Transactive memory system develop as team members learn about one
another’s expertise [9] accomplished predominantly through interpersonal communication [32]. The extent to which communication is functional, or task-relevant, should be positively related to members’ learning about one another and thus to transactive memory system [23]. When reviewing related literature, we found that promising laboratory experiments face difficulties when extending their results to work groups because experiments in which students performing a single task for a brief period are unlikely to capture real-world phenomena in work groups [44]. The tasks type and task structure can heavily influence transactive memory system benefits in work groups. Tasks used in laboratory experiments can artificially create the phenomenon of transactive memory system that may or may not exist in work groups. For example, transactive memory system has been measured by observing students as they assemble radios [30] and this is replicated [27], [35] to mediate different factors that could influence findings.

Current literature has examined wide range of groups, tasks and settings in transactive memory research, it assumes that a transactive memory system is useful for all types of tasks performed by groups in knowledge intensive organisations and assumes that groups perform a single “type” of task [45]. Yet, there is evidence that transactive memory system may not lead to the same performance effects for all types of tasks [36], [42]. Complex and dynamic tasks that some work groups face daily could have a large impact on transactive memory system development and structure [44]. Therefore, instead of featuring artificially imposed knowledge domains in laboratory experiments, the functioning of ongoing work groups is likely to depend on the pooling of overlapping, complementary and unique knowledge [38]. Therefore, when measuring a transactive memory system, careful attention should be paid to the type of tasks and knowledge domain in which a team is performing the task. Tasks can be categorized in terms of three elemental processes (produce, choose, and execute) – upper branch in Fig. 2. and three structural qualities of the task – lower branch in Fig. 2., relating to task demands (divisible versus unitary), the underlying goal structure of the task (cooperative versus conflictual), and the evaluative specificity of group outputs (intellectual versus subjective) [45]. Task structure and level of benefits that team with well-developed TMS receives from each type of task are shown in Fig. 2.

![Figure 2. Task structure and benefits.](image)

Based on the above description of the features and benefits of a well-developed TMS, the tasks that are likely to benefit from a TMS are those for which performance depends on access to diverse knowledge, access to deep and specialized knowledge, access to credible and correct knowledge, awareness of which members possess what expertise and need to apply large amounts of knowledge to solve the task, possibility to share workload for the task, combining and integrating members’ knowledge and cooperation to complete the task, efficient coordination of members’ activities and new learning that during task processing [45].

### 2.4 Research gaps

In summary, to the best of our knowledge, although there are two studies on transactive memory systems in software engineering, the question of how transactive memory systems are impacted specifically in software teams in large-scale projects has not yet been studied.
We need to gain more understanding of how knowledge is managed in and across teams and how different knowledge is flowing through an organization. To develop a better understanding if transactive memory systems have same effects on team performance in large-scale software development, we will see how different software practices, different management decisions about team setup (cross-functional teams, function teams), work environment and in-team agreements influence teams’ performance from a transactive memory system perspective.
3 RESEARCH APPROACH

3.1 Research methodology

As described earlier, this thesis work is a part of a research project TEDD and a larger empirical investigation, which follows a multi-case study methodology. To answer our research questions, we were able to influence the design of some empirical data gathering techniques, participate in parts of data collection and transcribe and analyze some of the already collected (recorded) data. Our study is thus can be characterized as an empirical exploratory multi-case study that is based on a survey and focus groups. The results of the research were used to write one research paper (For Paper-1 see Appendix I). In the following we explain the choice of research methods and given an overview of research activities.

Our related literature review revealed that there are different research methodologies, including case studies, experiments, factor analysis and surveys, which are available to empirically investigate the proposed research on transactive memory systems. Each method has its suitability for a specific situation or requirement. It is the responsibility of the researcher to decide which technique is appropriate in which situation. As in our research we choose to explore whether transactive memory systems are applicable in a new context (large-scale development), we decided to design our research study as exploratory multi-case study. As we were provided access to two companies for conducting the study, it follows the multi-case approach which means that the study results are more generalizable. Controlled experiments have a greater control on variables and higher internal validity, but would be unable to capture such large and complex context. However, our decision to use case study research design has somewhat mitigated this threat.

We also decided to use a survey as it is a common comprehensive method for gathering empirical evidence in social sciences and its use is recognized in software engineering [46]. Systematic collection of information through standardized questionnaires is the major source of relation data [47] such is social networks. Survey instrument has a good external validity and can easily be generalized as it represents the opinion of the population from different real-world contexts [48]. To support this and probably compare data in the future work, we decided to replicate the survey from previous research [43], with a few extensions. This also serves for the assessment of the reliability and precision of the survey and also prevents us from unknowingly duplicating previous research.

The following table represents all research activities conducted by the thesis author during the study.

Table II. Study design

<table>
<thead>
<tr>
<th>Activity</th>
<th>Timeframe</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial literature review on TMS</td>
<td>November 2013 – January 2014</td>
<td>Read around 20 papers. Gained knowledge how TMS works and why it would be applicable in software development.</td>
</tr>
<tr>
<td>Activity Description</td>
<td>Dates</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Participation in transcribing Company A main site Focus groups with four teams</td>
<td>December 2013 – January 2014</td>
<td>Transcripts accounting 68+ pages</td>
</tr>
<tr>
<td>Participation in interviews in Company B other Swedish site</td>
<td>February 2014</td>
<td>Three interviews and one focus group.</td>
</tr>
<tr>
<td>Observations in Company A main site</td>
<td>February 2014, October 2014</td>
<td>Participation in TEDD meetings of outgoing research</td>
</tr>
<tr>
<td>Participation in focus groups in two Company B sites in Sweden</td>
<td>February 2014</td>
<td>Two focus group interviews.</td>
</tr>
<tr>
<td>Observations in Company B main Swedish sites</td>
<td>February 2014</td>
<td>Observation notes of project.</td>
</tr>
<tr>
<td>Transcribing Company B focus group with three teams</td>
<td>March 2014</td>
<td>Transcripts accounting 71 pages</td>
</tr>
<tr>
<td>Conduction of a focus group in Company B offshore site in India</td>
<td>July 2014</td>
<td>One focus group interview.</td>
</tr>
<tr>
<td>Conduction of interviews in Company B offshore site</td>
<td>July 2014</td>
<td>8 interviews with developers and managers.</td>
</tr>
<tr>
<td>Observations in Company B offshore site</td>
<td>July 2014</td>
<td>Observation notes of offshore site.</td>
</tr>
<tr>
<td>Systematic literature review (in transactive memory system)</td>
<td>January 2014 - May 2014 – as part of Research Methodology course</td>
<td>Identified 22 relevant studies of TMS.</td>
</tr>
<tr>
<td>Review of related research literature (large-scale software development, knowledge management, social-network analysis)</td>
<td>September 2014 to March 2015</td>
<td>Extracted data of measurements and benefits of TMS from 22 studies.</td>
</tr>
<tr>
<td>Initial data analysis</td>
<td>February 2015</td>
<td>Data clarification with companies and analysis with SNA tools.</td>
</tr>
<tr>
<td>Data analysis</td>
<td>March-December 2015</td>
<td>Data analysis of survey and focus groups.</td>
</tr>
</tbody>
</table>
The research started with an initial review of related literature. In order to build our research background, we performed a literature study by using snowball sampling technique [51]. We used backward snowballing (references) from literature review by Lewis [45]. This phase provided us with the basic information about transactive memory system mechanisms and possible benefits from the following studies (See Appendix VI). It also provided the base to transactive memory measurements and construct for survey and interviews.

This empirical research uses combined approach that include both quantitative and qualitative data collection and analysis. Quantitative data is used for measuring transactive memory system. Qualitative data will be used for analysis of project characteristics and team and task characteristics and practices that support development of TMS. Quantitative data for this research is collected using a questionnaire. Qualitative data for this research is collected by focus groups and interviews. Individuals and teams were asked directly about their ways of working, connections, affiliations that are necessary for given software development task. This technique is widely used in software engineering research [46]. While the author has participated in some focus groups, the main contribution of the author lies in designing, administering and analyzing the data from the survey, which is in more details described in the next section.

### 3.2 Survey

#### 3.2.1 Survey questions

To answer our research questions we have conducted a combined transactive memory system and social network analysis survey to measure and capture transactive memory relations of the team members inside and outside of the team, and to understand why these relations occur and what are their consequences. We used a latent variable model to measure transactive memory processes and analyze transactive memory structures using social network analysis techniques [49], [50]. The design of the survey is a partial replication of an empirical survey by Manteli et al. [43]. We extend it by obtaining a directed knowledge network and acquiring detailed information about each knowledge-sharing relation. Also, in contrast to Manteli et al., who applied the survey on the project and unit level, we trace our observations to the team level. The transactive memory system survey is available in Appendix II.

We asked respondents to identify in “free-recall” format those with whom they exchange knowledge that is related to tasks they do, to identify persons whom the respondents transfer the project-related knowledge to, or retrieve the knowledge from, as well as describe the nature and content of the knowledge transferred or retrieved. These could cover survey respondents and also non-respondents (members of other teams or supporting roles). The respondents were then asked to evaluate the identified relations using a 5-point Likert scale for the following transactive memory system characteristics:

- Awareness of the knowledge needs (knowledge transfer)
- Awareness of the expertise (knowledge retrieval)
• Frequency of transferring the knowledge sharing (knowledge transfer and retrieval)
• Availability of a contact (knowledge retrieval).

In result, we obtained transactive memory scores in dyadic level and could aggregate them to team level, distinguishing between scores within contacts of team members and within contacts of other team member or supporting roles. We also have obtained a knowledge network, i.e. indicating whom and what kind of knowledge they transfer or receive. This network includes not only participants of the survey, but also non-participants recalled by the survey respondents, such as team members who did not participate, members of other teams or supporting roles. We further support and triangulate our results by means of interviews and data clarification.

3.2.2 Survey sampling
Sampling technique used was convenience sampling – selecting most accessible subjects to author [61]. The sampling was done with the help of the company representatives. The teams were selected (1) from a variety of two sites in Company A and from three sites in Company B; (2) in both companies’ teams were selected both new and mature and teams working with familiar and unfamiliar tasks (3) teams had 5 to 9 members.

3.2.3 Survey execution
We conducted a survey with the all selected teams from both projects participating in the focus groups. The survey was web-based in Company A for the Swedish sites (teams A, B, I and P) and was done four months after focus groups and handed on the paper printouts for the offshore site (teams G, N, R) and was done right after focus groups. The survey was handed on the paper printouts for the Swedish sites (teams I, C and S) and was done right after focus groups and was handed on the paper printouts for the offshore site (group I) and was done right after focus groups. To conduct the web-survey the author of this thesis created a website and prepared each team member an individual invitation to complete the survey. Each e-mail contained detailed instructions (see Appendix III) on how to complete the survey and how and which connections to report. In cooperation with company representatives it was done in time when workload was smaller and participants were given two days to fill surveys. When handing out the paper printouts participants were given detailed instructions on how to complete the survey and those instructions were clarified if necessary during the time of completion. Participants were given 30 minutes to fill in the survey.

3.2.4 Response rate
In total members of 11 teams participated in the study for a total of 81 participants. Actual team sizes ranged from 5 to 9. Not all members of all teams completed study, so we included teams with three or more respondents, to comply with other TMS studies [1] resulting in a usable sample of 49 members and 9 teams (see Table III):
1) Company A: In total, 40 people from 7 teams completed the questionnaire. However, one of the teams was removed from the analysis due to a very low response rate (33%). The response rate for the remaining 6 teams in total was 90% and for individual teams did not fall below 71%. Final sample consisted of 35 project members.
2) Company B: In total, 26 people from 4 teams completed the questionnaire. However, one of the teams was removed from the analysis due to not formed as a team. The response rate for the 3 teams in total was 88% and for individual teams did not fall below 67%. Final sample consisted of 14 project members.
Table III. Sample and response rates

<table>
<thead>
<tr>
<th>Teams</th>
<th>Location</th>
<th>Team members in total</th>
<th>Participated in the survey</th>
<th>Response rate</th>
<th>Included/excluded from study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team A</td>
<td>Sweden</td>
<td>8</td>
<td>6</td>
<td>75%</td>
<td>Included</td>
</tr>
<tr>
<td>Team B</td>
<td>Sweden</td>
<td>7</td>
<td>5</td>
<td>71%</td>
<td>Included</td>
</tr>
<tr>
<td>Team I</td>
<td>Sweden</td>
<td>6</td>
<td>2</td>
<td>33%</td>
<td>To few responses</td>
</tr>
<tr>
<td>Team P</td>
<td>Sweden</td>
<td>6</td>
<td>5</td>
<td>83%</td>
<td>Included</td>
</tr>
<tr>
<td>Team G</td>
<td>China</td>
<td>9</td>
<td>8</td>
<td>89%</td>
<td>Included</td>
</tr>
<tr>
<td>Team N</td>
<td>China</td>
<td>5</td>
<td>5</td>
<td>100%</td>
<td>Included</td>
</tr>
<tr>
<td>Team R</td>
<td>China</td>
<td>6</td>
<td>6</td>
<td>100%</td>
<td>Included</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td>81</td>
<td>63</td>
<td>78%</td>
<td></td>
</tr>
</tbody>
</table>

3.2.5 Avoiding Bias

To avoid bias that the survey wouldn’t represent actual situations in companies, we carefully tried to eliminate the bias at this stage of our study. We followed-up non-response to confirm that there is no systematic bias on the results due to non-response that would make identification of structural features impossible in sample set of data [63]. We used the following follow-up plan for our survey study:

- We sent them reminders,
- We contacted company managers to update on the response rate and clarify if there were any specific reasons for non-responses,
- Contacted one-by-one through email reminders.

3.2.6 Survey data preparation

We conducted survey data preparation to ensure reliability and quality of the collected survey data [63]. Data screening and data transformations of the individual responses was performed. That included the following steps:

- Clarifying the names and roles of each network contact. Some names were spelled differently, some had only name, especially by remote colleagues and some were hard to read. We also requested the managers of both companies to clarify names and each contact’s role as we had large set of non-participants in the contact lists, including both formal and informal roles. After that, we merged survey responses in the final set of data.
- Clarifying unclear responses (e.g. unknown abbreviations of roles, processes and sub-systems) were identified and follow up emails were sent to the company representatives for clarification.
- Removing invalid responses – these were responses, in which e.g. the knowledge field was not empty, but contained a clear message that it should be empty (“not much” or “I am new in the team and I only receive knowledge from others”).
- Merging reciprocal relations. Reciprocal are relations, in which Person A identifies Person B as a knowledge-sharing contact, and Person B identifies Person A in return. Such relations were identified and merged by semantically combining the knowledge content fields and computing the average values for the characteristics of the
connection by formula (respondent A -> connection B; respondent B -> connection A => for knowledge exchange fields (A transfer to + B receive from)/2 => (B transfer to + A receive from)/2 and (A + B)/2 for rest of data fields). We do not report the amount of reciprocal relations, since the resulting networks include both survey respondents and non-respondents. The end result is a set of data that represents who communicates with whom, and a set of six valued relationships for every connection.

- Finally, data transformation was necessary to reflect the knowledge flows, i.e. directed connections in terms of incoming, outgoing and exchange flows between two contacts in the network, instead of the direction of responses in the survey, i.e. who referred to whom.

### 3.3 Data Analysis

The results reported in Paper-1 focus on finding whether the teams in large-scale software development can benefit from a well-developed transactive memory system. Further, we were interested to see how different factors as task complexity, task familiarity and team knowledge networking practices and different management decisions about team setup (cross-functional teams, function teams) affect the teams transactive memory system and performance. This data analysis that helps to answer the research questions is detailed in the following sections.

#### 3.3.1 Survey data analysis

We analyzed survey data in two ways. First, we collected social network data allowed us to express transactive memory system both as a dyadic relationship between two people, as well as a team level relationship. We also distinguished transactive memory system relationship outside team. As we used a partial replication of an empirical survey by Manteli et al. [43] we also decided to use same formula for calculating dyadic relationships. This survey based on latent variable model was empirically validated and Cronbach’s alpha coefficient for reliably was calculated at. 708 [43]. Indirect way of measuring TMS is guided by the three transactions of directory updating (DU), information allocation (IA) and retrieval coordination (RC). Directory updating was constructed using one item (Q1), information allocation was constructed using items Q2 and Q5 and finally retrieval coordination is constructed using items Q3, Q4 and Q7. We calculated TMS at the dyadic level according to Eq. (1).

\[
TMS_{dyadic} = \frac{Q1 + \mu(Q2 + Q5) + \mu(Q3 + Q4 + Q7)}{3}
\]

We calculated TMS at the team level according to Eq. (2). N is number of dyadic TMS connection each team member has in team. Similar formula was used to calculate TMS at the project level. To calculate TMS at the project level we used connections reported with other teams and supporting roles.

\[
TMS_{team} = \frac{\sum_{x=1}^{\text{number of team members}} TMS_{individual_i}}{\text{number of team members}},
\]

where \(TMS_{individual_i} = \frac{\sum_{j=1}^{J} TMS_{dyadic(i,j)}}{N}\)

We calculated TMS scores on dyadic level and then averaged on team level, distinguishing between Team’s internal TMS and team’s external TMS. Team’s internal TMS is perceived
level of TMS between team members, while team’s external TMS is reported level of TMS from team members to external contacts. Overview of team’s internal TMS and team’s external TMS for each team with both performance measurements and company scores are presented in Table IV.

<table>
<thead>
<tr>
<th>Team</th>
<th>TMS&lt;sub&gt;internal&lt;/sub&gt;</th>
<th>TMS&lt;sub&gt;external&lt;/sub&gt;</th>
<th>Effectiveness</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team C</td>
<td>4.03</td>
<td>3.84</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>Team S</td>
<td>3.86</td>
<td>3.46</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>Team I</td>
<td>3.70</td>
<td>3.74</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td>Team A</td>
<td>4.33</td>
<td>3.76</td>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>Team B</td>
<td>3.98</td>
<td>3.42</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Team P</td>
<td>3.98</td>
<td>3.85</td>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>Team R</td>
<td>3.79</td>
<td>3.80</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>Team G</td>
<td>3.63</td>
<td>3.75</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>Team N</td>
<td>3.86</td>
<td>3.30</td>
<td>21</td>
<td>8</td>
</tr>
</tbody>
</table>

### Table IV. TMS scores within and outside team

<table>
<thead>
<tr>
<th>TMS</th>
<th>M</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company A</td>
<td>Internal</td>
<td>3.88</td>
</tr>
<tr>
<td></td>
<td>External</td>
<td>3.67</td>
</tr>
<tr>
<td>Company B</td>
<td>Internal</td>
<td>3.92</td>
</tr>
<tr>
<td></td>
<td>External</td>
<td>3.37</td>
</tr>
</tbody>
</table>

3.3.2 Team performance data analysis

We asked company managers to evaluate team performance. We adapted questions used in a similar context by previous research [11]. We asked project managers from both projects to assess how well each project team performed compared to other software teams with which they were familiar, on dimensions such as work quality, team operations, ability to meet project goals, extent of meeting design objectives and reputation of work excellence [52]. – see Appendix V for questionnaire. We did send questionnaire and instructions by e-mail and further clarified them, if requested by project managers. We averaged the five items to develop a measure of team effectiveness (alpha = 0.86, mean = 4.07, s.d. = 0.75)

The other essential dimension of performance that software teams are regularly measured on is efficiency, which is usually measured by project cost and time-to-completion [7]. We asked two separate questions about the team’s adherence to schedules and budgets and created an efficiency measure by averaging the stakeholders rating on these two (except on one site in project of Company A, where was only time-to-time completion measured) items (alpha = 0.80, mean = 3.39, s.d. = 0.99).

<table>
<thead>
<tr>
<th>Team</th>
<th>Work quality</th>
<th>Team operations</th>
<th>Ability to meet project goals</th>
<th>Extent of meeting design objectives</th>
<th>Reputation of work excellence</th>
<th>Adherence to schedule</th>
<th>Adherence to budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team C</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Team S</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Team I</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
3.3.3 Correlation and regression analysis

The selection of an appropriate statistical test is based on study design, research questions/hypothesis and the characteristics of the data. As in our case we know that our data are quantitative and parametric and group of samples is one group. We then used correlation techniques to measure statistical relationship between TMS_{internal} score and performance measurements.

We found that the team (inside) TMS composite was weakly related to team effectiveness ($r = 0.48, \ p < .01$) and team efficiency ($r = 0.47, \ p < .01$). Correlation scores can be found in Table VI. As we suspected, only team’s internal TMS score doesn’t explain team performance in large-scale distributed software development projects.

Table VI. Team TMS and performance correlation

<table>
<thead>
<tr>
<th>Variable</th>
<th>$M$</th>
<th>$SD$</th>
<th>$1a$</th>
<th>$1b$</th>
<th>$2a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study (N = 9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1a. Internal TMS</td>
<td>3.91</td>
<td>0.20</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b. External TMS</td>
<td>3.65</td>
<td>0.20</td>
<td>0.13*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2a. Effectiveness</td>
<td>20.22</td>
<td>3.77</td>
<td>0.48*</td>
<td>-0.10*</td>
<td>-</td>
</tr>
<tr>
<td>2b. Efficiency</td>
<td>6.78</td>
<td>1.99</td>
<td>0.47*</td>
<td>-0.45*</td>
<td>0.66*</td>
</tr>
</tbody>
</table>

Significant at $^*p < .01$

We used trend line plot to identify relationship graphically and to identify outliers. For correlation chart, see Figure 3.

Figure 3. Correlation trend line plot.
Furthermore, we used T-tests to test for statistical significance of our findings. T-test are common when set of data consists of one sample as well as interval and ratio level data [53]. T-test has following assumptions which coincide with our data - data is normally distributed and data has random selection of sample from population.

Also, as part of analysis we checked the content of the knowledge field and learned that respondents reported very different types of information as the knowledge of their job. Author participated in creation of coding scheme of this knowledge and partially participated in classifying the responses in following categories:

- product-related knowledge - knowledge about software product parts, design and architecture.
- process-related knowledge – knowledge about ways of working and use of development tools, tips and tricks of development.
- work coordination or administrative coordination - formal policies and procedures, project milestones and delivery schedules, requirements and design specifications

We used this data to further support our findings about team characteristics from focus groups.

![Figure 4. Influences on TMS and performance](image)

We used regression analysis to explore the influence of other factor on team performance. We used this technique to follow similar analysis as used in the TMS research [33] [38]. We examined the relative effects of team’s internal TMS and other control variables, namely team’s external TMS (with the rest of organization) and the number of unique contacts per team member, on overall performance using a standard multiple regression analysis. We propose hypothesis regarding the impact of having access to differentiated knowledge from other teams and experts, and ability to use this expertise pool in a well-coordinated fashion on performance (see Figure 1, blue boxes). Variable of the number of contacts per team was averaged by the number of team members in the team for two reasons: that better represented factor we were investigating (efficiency of outside network of team), that was in compliance with TMS scores, that were averaged by the number of team members in the team.

First, we examined the relative effects of team’s internal TMS and other control variables, namely team’s external TMS and the number of unique contacts per team member, on overall performance using a standard multiple regression analysis. We created two models - that team’s internal TMS score and team’s external TMS score will positively predict perceived effectiveness of team by external evolution (H1) and that team’s internal TMS score number of unique contacts team have will have positive influence on perceived effectiveness of team.
by external evolution (H2). Further, we created a model that all three mentioned variables together will predict team performance. Thus, our hypothesis is that in large-scale software development context teams with high score of combination of team’s internal TMS, team’s external TMS and number of unique contacts team have will have positive influence on perceived effectiveness of team by external evolution (H3).

The results of the regression analysis are provided in the Paper-1 (Section IV - Results). The traditional factors model had an adjusted R2 of 0.00 for H1 and R2 of 0.61 for H2 (Significant F < 0.01, all variable levels under p < .05). Thus both of these hypotheses were not supported.

The traditional factors model had an adjusted R2 of 0.77 (Significant F < 0.01, all variable levels under p < .05, ) and thus was a significant predictor of team effectiveness.

3.3.4 Focus group data analysis

We used focus group data analysis to understand the context in which teams were operating. Our literature review shows that TMS have different implications on different types of teams and task types. Our literature review in large-scale projects and initial observations from companies suggest that a team may have different needs for knowledge networking. Qualitative data collected through focus groups are then analyzed by using the thematic coding [62] which helped to identify, analyze and report the patterns in the data during its six phases. These six phases are: understanding the data, generating initial codes, searching themes among codes, reviewing themes, defining and naming themes and writing the report. Data gathered through interviews is used to support our decisions while working on survey data analysis. We followed this by creating coding scheme based on TMS survey [23]. The codes were then read and grouped under more general categories. The categories were grouped again under more generic topics. The codes under each category, topic-wise, were read separately and a summary was written for each category. Categories were:

- Cross and within team collaboration
- Process issues
- Knowledge coordination

From our literature review of TMS we also identified factors that may influence well-developed TMS in teams. We also collected information about the following items:

- Team type,
- Team stability,
- Number of part-time members,
- Task complexity,
- Task interdependency with other sub systems,
- Task familiarity,
- Reliance (complex tasks),
- Reliance (familiar tasks),
- Participations on Communities of practice.

The author of this thesis wasn’t able to quantify the results of this research step due to limits of available data and thus only observations of possible implications of these factors on the development of TMS were made.

3.4 Results
In this chapter we summarize our results following the research questions. For more detailed results, see Paper 1 (Section IV – Results).

RQ1: How does a well-development TMS influence team performance in large-scale development?

We have identified that generally well-developed TMSs in large-scale distributed development correlates with performance, as promised by studies from other knowledge-intensive field settings. However, we learned that while there is a positive relationship there are multiple additional factors that may have an influence.

Our results from correlation analysis between team’s internal TMS and perceived performance by project managers show that a good team’s internal TMS alone may not be sufficient to have good team performance in large-scale software development. We tested our hypothesis that team’s external TMS and teams’ member unique contacts per member are associated with team performance. Our regression results showed that this hypothesis is supported. This indicates that in large-scale setting team needs not only a team’s internal TMS, but also sufficiently large and efficient network that fulfills team’s knowledge needs.

We investigated few of those factors that we identified from previous research on TMS and knowledge management as important. We identified that complexity that creates large-scale distributed development settings in knowledge communication and coordination with other teams and supporting roles, team’s external TMS should be developed and teams needs network of unique contacts that is effective to fully benefit from well-developed TMS.

This supports our findings that that a single team cannot posses all the necessary knowledge within the team and developers needs years to gain good understanding of developed system. We found that this usually leading to some specialization in certain area or specialization in module and using supporting role expertise to gain the system knowledge on the higher level. While some of those practices seems support development of TMS, others may hinder it, we discuss this in more detail in answer of RQ2. Furthermore, knowledge network of teams and experts should be efficient as it requires time to build connections and understanding of each others expertise and time is needed to communicate and maintain this network. Thus, team members should have some level of specialization not only in knowledge, but also in sharing networking burden.

RQ2: What influences the development of a TMS in large-scale software development?

We use our qualitative data analysis from focus groups to answer this research questions. We couldn’t perform correlation or regression analysis because we weren’t able to quantify those other factors that may influence developed of TMS and benefits of TMS.

From the use of previous studies of TMS we identified that task types and team processes that may have influence of on development of TMS. We used focus group transcript coding scheme to identify influence of those factors and then we compared our focus group analysis with both TMS scores. We identified teams with well-developed TMS and teams with weakly developed TMS and analyzed practices that may have caused development or hindrance of development of team’s TMS.

We identified that teams that received more complex tasks with more task interdependency and cooperative nature showed better TMS scores comparing to teams that received tasks that could be divided and executed alone and was divided by the team leader (in one case). Team’s knowledge needs increase with the growing task complexity and task interdependency, and when the team works on unfamiliar tasks.

We found, as previous research suggests, that both functional and cross-functional teams can develop good TMS and benefit of well-developed TMS if they receive beforehand mentioned tasks. Team stability also affected TMS, were teams with better stability tend to have better
TMS. Teams that had better team cohesion, discussions about solutions, giving feedback and relying on each other also showed better developed TMS and better performance scores. Thus, our findings suggest that managers of software development companies and teams in software development should be aware of practices that may help or hinder development of TMS, both, within teams and with other teams and supporting roles.

3.5 Discussion

In this chapter we summarize our discussion following the research results. For more detailed discussion, see Paper 1 (Section IV – Discussion).

3.5.1 TMS in large-scale software development projects

Our findings suggest that both for teams and their team members establishing, maintaining and extending efficient knowledge network time and effort is needed. Recently assembled teams had lower team’s external TMS scores, especially when they did put emphasis on team’s internal TMS development. Thus efficient structures to identify and to communicate with other teams, experts and supporting roles should be established. One way how to do that, as our findings suggest is use TMS and TMS practices to understand knowledge needs of teams. Large-scale software development development project managers can improve their practices on team assembly and task allocation to support teams’ TMS and have benefits on team performance from well-developed teams’ TMS. We suggest that task should be allocated so that teams are able to cooperate on solution, or at least make decisions together on how to allocate tasks. That would help develop team TMS. For example, as our results showed that teams that had greater discussions within team about design and solutions on tasks had better developed TMS.

Our findings suggest that in teams in large-scale software development project should be aware of practices that positively influence their performance. Teams with better cohesion, familiarity and teams that solved task together and relied more on networking both within team and outside team had better perceived performance results. Teams that had strategies to introduce newcomers with helping solve tasks by team discussion and providing feedback also had better developed TMS. However, we learned that teams are not always aware of these practices, so project managers should provide help.

Our approach on how to examine knowledge networks in large-scale software development can be used by both practitioners and researchers as a cause and effect tool for understanding the status of TMS of their teams and using knowledge network data to improve coordination and collaboration between teams and supporting roles of large-scale software development.

3.5.2 Implications for Practice

There are several practices from our results that is supported by previous research that would be applicable in software engineering for managers of large-scale projects. These practices are attributed to training teams, assembling teams, task characteristics and communication. Previous research shows that teams that have longer experience working together Error! Reference source not found. request and accepted backup from one another more. Our results show that in both organizations teams that had longer experience working together had better developed TMS. We therefore, suggest that keeping team together for longer time will help those teams develop TMS and have better team performance on long run. Previous research on TMS suggest that groups that train together develop better TMS [27]. From our results we suggest that managers can support team’s internal TMS development by challenging team with new and more difficult tasks, as it was especially seen in case of Team A.
Further, our findings suggest that task allocation strategy for teams should take account need for development of team’s TMS and need to benefit from well-developed team’s TMS. To develop team’s TMS, managers should allocate tasks so that teams receive cooperative and divisible tasks. These tasks will more likely motivate team members to learn what other team members know and further develop ways of strengthening the specialization in the team [45]. Previous research of TMS also shows that face-to-face communication have large role of developing TMS. Allocating part-time team members will also have negative effect on development of TMS. Allocating team members to multiple projects our task-force teams that requires spending time away from team will also have similar negative effect. For team members should have practices that facilitate face-to-face communication such as daily stand-up meetings will help develop team’s internal TMS.

Our results from regression analysis that team team’s external TMS and number of unique contacts team have will have positive influence on perceived effectiveness of team suggest that both managers and team members should be aware and support external networking processes. These results were further supported by focus group results that most of teams rely on networking, when are required to solve complex or unfamiliar tasks. This implicates that managers should work on creating a network of available experts and supporting roles for team members to gain the necessary knowledge to solve their tasks. Managers should acknowledge and support need to spend more time to networking when allocating complex and/or unfamiliar tasks. For managers this implicate that they should support team communications needs with allocating time to CoP and creating enough work-space for meetings for teams and between teams. For team members that would mean that dividing who will attend and attending CoP and other similar meetings will help develop team’s external TMS.

3.5.3 Validity Threats

The study has few limitations of reliability and generalizability. The first reliability of conclusions limitation is that the empirical data related to the social network are self-reported. In other words, the study investigated the perception of the network and did not attempt to calculate or determine an objective measurement of the social network. Consequently, knowledge and information sharing could have been exaggerated or underestimated by the responses.

The second reliability of conclusions limitation is sampling strategy. Author was accessible those companies and teams that was part of TEDD project. While it allowed access to otherwise unavailable context – large-scale distributed development it limited ability to use more rigorous technique of sampling in terms of transactive memory system research to limit control variables in teams’ characteristics. We mitigated this by combining case study and survey approach.

The third reliability of conclusions limitation, we used partial replication of questionnaire by Manteli [43] and it had Cronbach’s alpha coefficient of their data set calculated at .708 it is has limited set of questions about each construct of latent variable model to measure TMS processes compared to original latent variable model developed by Lewis [23]. We choose to use this questionnaire to gain data about TMS structure and due to limited allocated time available from involved companies. This may have limited our ability to measure correctly TMS to unknown effect and threats to construct validity. Further effort to re-evaluate, validate and potentially improve questions used for the operationalization of the model is needed.

The forth reliability limitation is use of subjective rather then objective performance measures is a potential limitation of reliability. In future, researchers may want to collect more objective measures of performance. However, that has been proven as difficult in software development context.

The first generalizability limitation of study findings is an issue because our sample empirical data was collected from a limited number of teams and within two project and thus may not
be sufficiently representative of teams developing software at other organizations. Our observations might be influenced by particular and distinctive characteristics of the projects, such as culture, development methodology and project characteristics. Other studies, that will gather empirical data within similar conditions would be necessary in order to dispelling this threat to external validity.

The second generalizability limitation is that complex and real work environment limits our ability to control variables that may influence team performance in such settings. Further studies about complexity of this environment is needed to eliminate the threat to external validity.

The third generalizability limitation is that not all data used in this research was collected especially for this research and RQ. To be able to collect sufficiently large sample of data for research, author needed adapt some of already collected data for study (focus groups, interviews and observation notes of Company A site in Sweden). This partly limited use of those data for identifying practices that support well-developed TMS. Second, survey for those teams was done four months after focus groups, while other teams done survey right away after focus groups. Thus, limiting ability to compare Company A site in Sweden focus group data with survey.

The forth limitation is that real networks are not static, but rather dynamic. Relationships between people may change and communication networks evolve over time. In current study we examined only a snapshot of the networks. Future research on to compare transactive memory systems over time and observe their evolution would be interesting approach.
Our main conclusions of our study are threefold. First, we conclude that lessons learned from TMS research can be applied in large-scale distributed development. We conclude that here is link (affect) between the team formation and task allocation practices, and the development of teams’ TMS. This suggest future need for a longitudinal study when where we can measure the development of teams’ TMS based on the different organization practices. Second, we conclude that link between well-developed team’s internal TMS and team performance was mediated by team’s external TMS and efficiency of team’s external TMS. However, we conclude that other factors that our study was not able to quantify may have role on team performance as well, thus future studies to investigate role of those factors is needed. Third, measuring TMS in large-scale work setting is challenging. This is due to reason that large-scale context requires capturing very complex environment. To fully understand large-scale context, we need to capture TMS in two levels – team and organizational. To further understand of TMS in organizational level we would need to capture actual knowledge flows that are related to software development and work coordination.

Research on networks of different knowledge flows would yield additional understanding of how different knowledge is shared within organization. Comparing those networks with formal structures of organization would help identify weak points in organization work organization structure. Research on knowledge network characteristics in large-scale organization could yield implications on networking practices as distribution of contacts seems be more in form of “rich get richer”, i.e. few project members that have good communication skills and already large set of contacts tend to accumulate large part of network connections.

In our study we were not able to quantify all projects, team and task characteristics. Future research should focus on understanding large-scale software development project complexity. We identified two factors that also have influence on team performance, but we suspect that several other factors may also have influence on team performance. For example, we were not able to quantify data about part-timers, team cohesion, team practices, reliance on human, social, product capital of communities of practices that may also have influence on development of TMS and team performance. Also, management practices, such as allocation of expert and support role and their availability, development methods and documentation quality may have some role on team performance. Furthermore, certain individual attributes of the participants, such as the years of experience working in the company/project, might also influence on team performance. 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. Also, few teams in Company A had dedicated experts of system developed in their teams working part-time. This may have had influence on development of team’s TMS and accessibility of necessary expertise for solving tasks. Future research on comparing teams with dedicated experts in team may provide some insight on which strategy is better, locate part-time experts, part-time team member in team or full-time experts outside team.

There is need of research on knowledge networking characteristics to better understand benefits and drawbacks to different approaches of leadership and team member specialization and specialization strategies. We seek to understand connection distribution in network as our initial results suggest, that it follows factorial distribution, that is project members with good contacts tend to add even more contacts, creating very large communication network, i.e. “rich get richer”.

We would also like to extend use social network analysis to analyze knowledge networks within projects. That would help to understand differences between planned network and actual network, thus help companies adjust they practices to improve their knowledge flows.
within projects. It would also allow to understand different networking roles, such as bridgeheads, boundary spanners, critical actors and how some project members develop in those roles.

Also, real networks are not static, but rather dynamic. Relationships between people may change and communication networks evolve over time. In current study we examined only a snapshot of the networks. Future research on to compare transactive memory systems over time and observe their evolution would be interesting approach.

We would like further extend our approach to distributed software development and seek to understand whether well-developed transactive memory system have same benefits. In distributed software development awareness and accessibility is limited between sites, thus also possible development of TMS between sites. However, if we can use some of previous research of TMS to help facilitate management and communication strategies that develops TMS between sites, it may help teams achieve better performance on cross-site collaboration.
5 ACKNOWLEDGMENT

Author are sincerely thankful to all participants involved in our study for their interest and feedback. Author are especially thankful to thesis supervisor Darja Smite for guidance throughout thesis writing process and critical comments.
REFERENCES


APPENDIX I – PAPER
Benefits of transactive memory systems in large-scale development

Aivars Sablis

Abstract—Large-scale software development projects are those consisting of a large number of teams, maybe even spread across multiple locations, and working on large and complex software development. That means that neither the teams nor team members individually hold all the knowledge about the software being developed, and team have to communicate and coordinate their knowledge. Therefore, teams and team members in large-scale software development projects must acquire and manage expertise as one of the critical resources for high-quality work. This can be captured in a transactive memory system, a shared cognitive system for encoding, storing and retrieving knowledge between members of a group. In this paper we report our findings from collecting data from two large-scale distributed development companies with 9 teams in total. We use quantitative data from a survey and qualitative data from focus groups to measure and analyze transactive memory systems and their role in determining team performance. Our findings suggest that large-scale software teams can have a similar benefits as teams in other context from related research on transactive memory systems if transactive memory practices within the team are supported with networking practices in the whole organization.

Index Terms—Large-scale; software development; transactive memory system; empirical study; multi-case study; knowledge management; TMS; team performance; distributed; global software engineering; expertise coordination;

I. INTRODUCTION

LARGE-SCALE software development projects have become more common in recent years as software-intensive systems grow in size and complexity. Today, it is not uncommon that large companies have projects that have multiple teams and many supporting roles, involving in a project up to one hundred or more developers, experts and supporting roles. However, there are a lot of challenges related to large-scale development teams and many companies are still trying to increase the effectiveness and efficiency of large-scale software development projects [3][9]. Large part of this challenge is due to that software engineering is knowledge-intensive work [5].

Large-scale software development projects are challenging in multiple ways. Large-scale software projects are those consisting of at least two and coordination is achieved through forums [54]. Super large-scale software project is those with more then nine teams and coordination is achieved through forums of forums [54]. Large-scale projects are often distributed. Distributed software projects are projects consisting of multiple teams spread across multiple locations. Challenges to communication and coordination, control and finding expertise are issues that always have been formidable in large software engineering projects [7][9]. It is common that in large-scale projects systems developed are so large that none of team members can possibly possess all the knowledge about the project individually. The problem is even greater in distributed development [3] and in global projects [10]. Managers and engineers must deal with challenges of coordinating work across sites [8], and organizational problems. Fewer formal and informal interactions between teams’, which occurs in communication channels with less richness and psychological interaction can reduce teams’ awareness of others’ work tasks [12].

Software development is knowledge-intensive work and therefore human aspects, such as communication in a team and motivation of team
members of software development, are often more important than the technological aspects for better performance [13][14][15]. Thus, related work in theories of cognitive science could be particularly relevant in software development.

In this paper we focus on knowledge network characteristics and apply the findings from cognitive sciences, in particular, the theory of transactive memory system. Knowledge network characteristics in software development could have an impact on how we can manage knowledge in an organization and what type of actions we should take to improve performance. We suggest that we could use the theory of transactive memory system to understand what type of actions we should take to manage knowledge in large-scale software organizations. Transactive memory system (TMS) is the cooperative division of labor for learning, remembering, and communicating relevant team knowledge [1] and it is commonly described as “who knows what” in the team.

We explore TMS and knowledge management in large-scale software development in an empirical study conducted in two companies. We use both a quantitative and a qualitative approach. We collect quantitative data through a survey to measure teams’ TMS with a latent variable model. We analyze transactive memory structures using social network analysis techniques. Finally, we use focus group interviews to analyze management decisions with respect to team management, as a set of decisions based on two aspects; team structure and composition, and task allocation.

Our results suggest that a team working in large-scale development not only needs an effective internal teams’ TMS, but also needs a developed and effective external TMS network within the entire organization. Our results suggest that different management decisions regarding team assignments, team tasks, task allocation might have an impact on development of teams’ TMS.

The rest of the paper is organized as follows. In Section II we present the theoretical perspective and background. In Section III we the methodology of the study described in more detail and we introduce how research was conducted. Section III also present projects under study, giving background about teams and task complexity and contains our data analysis. Section IV contains resulting findings and Section V includes discussion about results and their implications. Section VI includes limitations of our study, discussion and future work.

II. RELATED WORK

A persistent question in general management research is competitive advantage and why certain firms excel in getting ahead while others fade away [19]. Organizational learning is a source of sustainable competitive advantage [19][20]. It is argued that organizations dynamic capability is one of arguments, which emphasizes a firm’s superior ability in sensing new opportunities in its environment and seeking those opportunities by continuously adapting, integrating, and reconfiguring its key assets and competences [22] and researchers have called for more work in this field [6][23]. The need for understanding how to manage knowledge in a software organization, a project, or a team has already been recognized, and in this regard the software engineering research community could learn much from the knowledge management community, which bases its theories on well-established disciplines such as cognitive science, ergonomics, and management [6][30].

In this paper we focus on what characterizes TMS in large-scale software development. As TMS is identified as a micro-foundation of dynamic capability for competitive advantage [21], we seek to understand whether benefits of TMS are similar in large-scale software development.

Transactive memory system is first theorized by Wegner [1] and could be described as "who knows what". That is, a transactive memory describes a social network of individual minds that transcends into a uniform agreement about the knowledge management both on an individual level and a group level [24]. This uniform agreement can be described best by three indicators of the manifestation of well-developed TMS in group [24][25]:

1) knowledge or memory specialization (the tendency for group members to remember different aspects of a task or to develop
specialized and complementary expertise),
2) task credibility (how much group members trust each other’s knowledge), and
3) task coordination (the ability of group members to work together smoothly and efficiently while performing a task).

Knowledge network characteristics in software development have an impact on how we can manage knowledge within such network and what type of actions we should take to get most benefits [27][56].

In the following sub-sections, we will motivate demands that large-scale development software development puts on knowledge management. We will motivate why and how TMS theory can address those demands and give pre-requisites for the development of teams’ TMS.

A. Knowledge management in software engineering

Knowledge management is a discipline that promises a gain in advantage on organizations’ intellectual capital. In general knowledge management is related to what actions management can take in order to support knowledge sharing, what some refer to as knowledge governance [6]. However, knowledge management is a large interdisciplinary field and as a consequence, there is an ongoing debate as to what constitutes knowledge management [6]. Commonly used definition of knowledge management is “a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company’s knowledge” [29]. In recent years, knowledge management in software development has also become a much-debated topic. Systematic review about knowledge management in software engineering [6] found that there are only few studies focusing on organizational and strategic aspects of software development. The three studies [16][17][18] in the organizational school that focuses on describing the use of organizational structures (networks) to share or pool the knowledge from, discuss the use of people networks in software organizations [6].

A software organization has many different needs related to knowledge and knowledge management. These needs can be viewed from business and knowledge perspectives. The main needs are to create a better quality software in shorter time and to make better decisions within software development process. This requires that software organizations create and use an enormous amount of knowledge to support business objectives with technology, process, project, product and domain knowledge in software development areas [28]. Software development is a process where technical and managerial decisions are made all the time. A software organization must foster and manage expertise as one of the critical resources for sufficient and high-quality work [2].

B. Knowledge needs

Software development is a group activity and many needs of knowledge management arise from that. Group members must communicate, collaborate, and coordinate. In large-scale projects were there are multiple teams, these teams also need to communicate, collaborate, and coordinate between them and with experts and supporting roles. Communication is often related to the transfer of knowledge. Collaboration is related to mutual sharing of knowledge. Studies of teamwork show how shared knowledge improves team effectiveness, because sharing knowledge about the development process and what’s to be developed helps teams avoid costly misunderstandings [4]. There thus is a need to collaborate and share knowledge. Coordination independent of time and space is facilitated if work artifacts are easily accessible [28]. A study of software development performance shows how better integration of domain and technical knowledge leads to increased software development effectiveness and efficiency [33]. Another study found that software developers apply just as much effort and attention to determining whom to contact in the organization as to getting the job done [32]. To achieve effectiveness and efficiency of knowledge sharing, software organizations need systematic ways of capturing knowledge and knowing “who knows what” [27].

Software organizations need to develop knowledge maps and identify sources of knowledge in terms of know-who and know-where [27]. Once such a
knowledge map is in place, it can be used to identify appointed and de facto experts, staff new projects based on skills and experience required, and identify knowledge gaps that indicate the need to hire new people or to develop training programs [27]. It is also found that communities built on existing networks are more likely to succeed [4].

C. Relevance of transactive memory systems research in studying knowledge management

Original definition of a transactive memory could be described by “who knows what” [1] and is a combination of individual knowledge and interpersonal awareness and uniform agreement of others’ knowledge [40]. When focusing on second part of transactive memory systems, the interpersonal awareness of others’ knowledge, as suggested by a tool developed by [24] there is the three basic processes of a TMS: directory updating, information allocation and retrieval coordination [1]. Directory updating 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. Information allocation in a TMS 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 the knowledge of the relative expertise of the individuals [41]. In other words, a well-developed TMS in group is a pre-requisite for effective and efficient knowledge management within a group. The better developed the group’s TMS is, the more the team can benefit from it.

When reviewing related research literature, we found that previous studies on TMSs show that organizations and knowledge teams benefit in several areas from well-developed TMS. Early studies of transactive processes show that well-developed TMS help members recall and discuss all available information and that well-developed TMS help to pool expertise that is distributed across members, at the same time increasing the chances that the group will find a solution or make a good decision on a choose task [54]. That has been observed in laboratory experiments with radio-kit assembly [38], [42], [46] and also in TMS in different field settings, such as top-management [47], and product development [48]. Large-scale and distributed knowledge-intensive work context partially is addressed in a sales company [45] and product development [50]. Knowledge intensive and stressful work is addressed in a study of air-traffic controllers [51] and also in a study of the positive impact of transactive processes on team’s creativity [52].

However, we found only two studies that were particularly addressing software engineering field. One study found that expertise coordination within teams affected some performance measures (effectiveness), but no others (efficiency) [2], with limitation to four cases and also called for additional studies. The other study found that different managerial decisions have different effects on TMS in a GSE setting [55].

These finding encourage future research of the benefits of well-developed TMSs in software engineering and an understanding of what impacts TMSs in software teams in large-scale projects.

D. Transactive memory system – knowledge network that we can measure

1) TMS measurements

For transactive memory analysis largely two types of measurements are used. Direct measurement [43] is used to investigate TMSs structure and indirect measurement is used to manifest TMSs existence in teams [41]. Latent variable model was developed by Lewis [24] and is used in several cases by other researchers [50] [52]. There are studies that use modified latent variable model [40] [52] as well as cases that used different independent variable models [45] [47]. In our study we use different latent variable model [55] because that would allow us not only capture TMS within team, but also capture knowledge network within projects. The actual measurement process is further described in the methodology section.

2) Performance measurements

The selection of performance measures is an important consideration in group research. In a study of software development teams [2], Faraj and Sproull found that expertise coordination within teams affected some performance measures but not
others. Thus, similar to that study we use two measures for performance: team effectiveness and efficiency. Both measures are assessed by the stakeholders. Team effectiveness is measured through evaluating five items: work quality, team operations, ability to meet project goals, extent of meeting design objectives and reputation of work excellence [38]. Team efficiency is measured through evaluating project cost and time-to-completion [39]. Although these evaluations are perception-based and are subjective in nature, we choose to use this measurements scale to be able to compare performance across teams and companies, since as objective measures of performance, such as lines of code per person month, often are subject to interpretation and manipulation.

3) Task types
When reviewing related research literature we have found that promising laboratory experiments face difficulties when extending their results to work groups. That is because experiments in which students performing a single task for a brief period are unlikely to capture real-world phenomena in work groups [55]. The tasks type and task structure can heavily influence benefits from TMS in work groups. Tasks used in laboratory experiments can artificially create the phenomenon of TMS that may or may not exist in work groups. For example, TMS has been measured by observing students as they assemble radios [42] and this is replicated in [38], [46] to mediate different factors that could influence the findings.

Current literature has examined a wide range of groups, task and settings in TMS research. It generally assumes that a well-developed TMS is useful for all types of tasks performed by groups in knowledge-intensive organizations and assumes that groups perform a single “type” of task [56]. Yet, there is evidence that TMS may not have the same performance effects for all types of tasks [47] [53].

Complex and dynamic tasks that some work groups face daily could have a large impact on TMS development and structure [56]. Instead of featuring artificially imposed knowledge domains, the functioning of ongoing work groups is likely to depend on the pooling of overlapping, complementary and unique knowledge [49]. Therefore, when measuring TMS, careful attention should be paid to the type of task and the knowledge domain in which a team is performing the task.

III. RESEARCH METHODOLOGY

A. Aim and objectives
In large-scale software development, where nobody can know everything, a team rarely possesses all the necessary knowledge. Even when the team specializes in a particular type of tasks or parts of the system, and achieves efficient and effective coordination and credibility within the team, their performance might depend on the knowledge they need to receive from outside of the team. Thus, in order to achieve the performance and to overcome the problems posed by the context of large-scale distributed software projects, effective expertise location techniques should be established within these companies.

As suggested by [6][27], there is a lack of empirical studies on knowledge management in software development. Also, software development industry could still benefit from team performance increase both in terms of effectiveness, ability to meet project goals and work quality [58] and efficiency measured as project cost and time-to-completion [59]. We need to gain more understanding of how knowledge is managed in teams and what are knowledge needs in large-scale software developed projects. This gap could be filled by applying cognitive science theories of knowledge management and group mind to software development. Thus, our aim is to fill these gaps by identifying whether well-developed TMS can provide the same influence and benefits in large-scale software development. To the best of our knowledge exactly how TMS is impacted specifically in software teams in large-scale projects has not yet been studied.

Thus, our proposed objectives are:

- Understand the type of knowledge networks and identify social network characteristics in large-scale software development companies.
- Identify whether well-developed TMSs in large-scale development leads to the benefits
in performance, as promised by studies from other knowledge-intensive field settings.

- Identify how different management decisions (such as team configuration, task allocation, support of team networking practices) affect the structure of TMSs as well as the processes developed within those systems in large-scale software development organizations.

B. Research questions

We want to understand whether teams of well-developed TMS have similar benefits in this context, because it has been regarded as promising in other knowledge-intensive disciplines. If an organization is aware of its existing TMS network, it can make decisions about how we use and develop further this network in most efficient way. Large-scale software development is complex environment and thus, multiple factors may have influence both on development of TMS and perceived performance of teams. Task allocation strategies, task characteristics and management decisions on project structure, team structure and composition may be most important. We also suggest, that for a team to be successful in large-scale development, team needs not only well-developed teams’ internal TMS, but also have sufficient contacts outside the team within the organization – both with other teams and supporting roles.

We seek to identify the link between large-scale development and organization management practices that improves software development team performance. Thus, we seek to address following

RQ1: How does a well-development TMS influence team performance in large-scale software development?
RQ2: What influences the development of a TMS in large-scale software development?

C. Research approach

To understand whether large-scale development also is a case where work groups benefit from well-developed TMS in terms of performance and whether we could apply the same team-building principles to stimulate the development of TMS in work groups, we choose to design our research study as exploratory multi-case study. In our research we use a combined approach with collection of both quantitative and qualitative data. We use qualitative data from the cases to understand what differences large-scale development has comparing to typical products as we suspect that large-scale characteristics of development bring additional complexity for a team TMS to work. We use quantitative data from a survey to measure teams transactive memory processes and their performance and also analyze what type of knowledge is exchanged in communication between team members and also outside of the team.

As we would like to capture both team TMS and organizational level TMS as well as identify external relationships [26] we will use "realist" approach as boundaries that are perceived as real by the participants and so correspond to the actual boundaries of social groups and organizations [57].

Qualitative data was collected by conversational techniques of formal and informal interviewing. Teams were asked directly about their knowledge connections necessary for their current software development tasks. Individual members of different management level were asked directly about project characteristics to clarify type and complexity of tasks in the studied projects and their involvement and connections, affiliations that is necessary for team software development tasks. We also used our results of interviews to further support and triangulate collected data from questionnaire. Data collection details are further elaborated in subsection F.

Quantitative data was collected using questionnaire. We decided to use survey as it is a common method for empirical evidence in social sciences and also is recognized in software engineering [34] and systematic collection of information through standardized questionnaires is a major source of relation data [35]. We measured a latent variable model to measure transactive memory processes. Our survey is a partial replication of a survey conducted by Manteli et al. [55]. Respondents where asked to describe the
knowledge received and transferred, as well as evaluate frequency of knowledge-sharing and accessibility of their contacts (using a 5-point Likert scale). However, we extend it by obtaining a directed knowledge network and acquiring detailed information about each knowledge-sharing relation.

D. Case selection
Sample of teams developing software were selected based on the study’s predefined requirements and was guided by company representatives: (1) teams were selected from a variety of two companies and different sites to maximise the scale in terms of geographic distance; (2) teams were selected both new and mature in both companies (3) teams had 5 to 9 members.

E. Empirical context
Data on software development teams were collected from the two different large-scale software development projects of the application development of the two large software development companies in Sweden. Company A specialized in telecommunication development; Company B specialized in automation technology development. Due to the growing complexity of their software products combined with the shortage of free resources and inability to employ in Sweden, both companies had distributed development of a number of software products across its sites in different locations spanning multiple time zones. The study was approved by company representatives from both companies.

F. Data collection
As a part of the multiple case study we have gathered empirical data from 11 focus groups, 62 survey responses and observations from visiting 3 different sites (all three in Company B). This study is part of a larger study that has gathered more extensive data, however the author reports here the data that has been used in the analysis. Parts of the data were gathered by other researchers, but transcribed by the author for the use in the analysis of the current research study. To improve the validity of our findings, we collected data through different means.

I) Observations
Onsite visits were organized to better understand the environments in the different sites of the two companies. The author has personally visited all sites in Company B and one site in Company A, and has used observation notes for the visits to all Company A locations, made by other researchers. The observations made during the visits were captured in the form of written notes.

2) Focus groups
Focus groups with selected development teams and groups were organized in each of the companies [60]. The focus groups followed a structured agenda where part of time was dedicated to acquire the information about the presence of the skills needed for solving team’s tasks, teamwork practices, interaction with other teams, roles and communities, usefulness of the externalized knowledge, teams’ reliance on different types of knowledge and skills in their daily work, and team’s perception of their performance and part of time was where part of time was dedicated to questions about TMSs where we followed questions from the survey developed by Lewis[1]. All focus groups in both companies were recorded and later transcribed. The author transcribed three focus groups that was conducted by other researchers with Swedish teams in Company A and participated in conducting and transcribing all focus groups in Company B. The majority of its team members represented each participating team, however some where unable to attend the focus groups due to leave or involvement in training.

3) Questionnaire
After focus groups a survey was organized in each of the companies. We conducted a survey with the eleven teams participating in the focus groups. The survey was web-based for the Swedish Company A site and handed on the paper printouts for the Company A offshore site, Company B Swedish sites and Company B offshore site. Web-based survey for Company A site was done few months later, but all other surveys were done right after the focus groups. Participants in paper-based surveys were given around 30 minutes to fill in as many questionnaires as they felt needed in “free-recall” format. Participants in the web-based survey were
given two days to fill in as many questionnaires as they felt needed in a “free-recall” format. Longer time was required due to the need of the participants to allocate time in their daily work schedule. We completed the list of recalled contacts and clarified these with company representatives to clarify the names, roles in the company and location.

In total members of 11 teams participated in the study for a total of 62 participants. Actual team sizes ranged from 5 to 8. Not all members of all teams completed the survey, so we included teams with three or more respondents, to comply with other TMS studies [24] resulting in a usable sample of 49 members and 9 teams (see Table I):

1) Company A: In total, 40 people from 7 teams completed the questionnaire. However, one of the teams was removed from the analysis due to a very low response rate (33%). The response rate for the remaining 6 teams in total was 90% and for individual teams did not fall below 71%. Final sample consisted of 35 project members.

2) Company B: In total, 26 people from 4 teams completed the questionnaire. However, one of the teams was removed from the analysis due to not formed as a team. The response rate for the 3 teams in total was 88% and for individual teams did not fall below 67%. Final sample consisted of 14 project members.

Table I. Survey participants

<table>
<thead>
<tr>
<th>Teams</th>
<th>Location</th>
<th>Team members in total</th>
<th>Participated in the survey</th>
<th>Response rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company A</td>
<td>Team A Sweden</td>
<td>8</td>
<td>6</td>
<td>75%</td>
</tr>
<tr>
<td></td>
<td>Team B Sweden</td>
<td>7</td>
<td>5</td>
<td>71%</td>
</tr>
<tr>
<td></td>
<td>Team P Sweden</td>
<td>6</td>
<td>5</td>
<td>83%</td>
</tr>
<tr>
<td></td>
<td>Team G China</td>
<td>9</td>
<td>8</td>
<td>89%</td>
</tr>
<tr>
<td></td>
<td>Team N China</td>
<td>5</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Team R China</td>
<td>6</td>
<td>6</td>
<td>100%</td>
</tr>
<tr>
<td>Company B</td>
<td>Team I Sweden 1</td>
<td>5</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Team C Sweden 1</td>
<td>6</td>
<td>4</td>
<td>67%</td>
</tr>
<tr>
<td></td>
<td>Team S Sweden 2</td>
<td>5</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td>57</td>
<td>49</td>
<td>86%</td>
</tr>
</tbody>
</table>

G. TMS and team performance: Team effectiveness and Efficiency

To measure team TMS we adapted questions used in a similar context by previous research [2]. We asked stakeholders to assess how well each project team performed compared to other software teams with which they were familiar, on dimensions such as work quality, team operations, ability to meet project goals, extent of meeting design objectives and reputation of work excellence [58]. We averaged the five items to develop a measure of team effectiveness (alpha = 0.86, mean = 4.07, s.d. = 0.75), similarly to what the authors in [2] have done.

The other essential dimension of performance that software teams are regularly measured on is efficiency, which is usually measured by project cost and time-to-completion [59]. We asked two separate questions about the team’s adherence to schedules and budgets and created an efficiency measure by averaging the stakeholders rating on these two (except in one site in the project in Company A, where we measured only the time-to-completion) items (alpha = 0.80, mean = 3.39, s.d. = 0.99). Team performance scores for all teams are reported in Table II.

IV. RESULTS

In this section, we present our findings regarding the characteristics of TMSs in two large-scale distributed projects at Company A and Company B. We start by presenting team performance scores and TMS composite scores for teams. We describe how work is organized at each project in the companies. We further discuss task type and how teams are set up and their knowledge needs when solving tasks. We use regression technique to analyze factors that influence team performance in large-scale networks.

A. Transactive memory scores

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 adapted [55]. We calculated TMS scores on dyadic level and then averaged on team level, distinguishing between Team’s internal TMS and team’s external TMS. Team’s internal TMS is perceived level of TMS between team members, while team’s external TMS is reported level of TMS from team members to external contacts. Overview of team team’s internal TMS and team’s external TMS for each team with
both performance measurements and company scores are presented in Table II.

<table>
<thead>
<tr>
<th>Team</th>
<th>TMSinternal</th>
<th>TMSexternal</th>
<th>Effectiveness</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team C</td>
<td>4.03</td>
<td>3.84</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>Team S</td>
<td>3.86</td>
<td>3.46</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>Team I</td>
<td>3.70</td>
<td>3.74</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td>Team A</td>
<td>4.33</td>
<td>3.76</td>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>Team B</td>
<td>3.98</td>
<td>3.42</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Team P</td>
<td>3.98</td>
<td>3.85</td>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>Team R</td>
<td>3.79</td>
<td>3.80</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>Team G</td>
<td>3.63</td>
<td>3.75</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>Team N</td>
<td>3.86</td>
<td>3.30</td>
<td>21</td>
<td>8</td>
</tr>
</tbody>
</table>

### Table III. Team TMS and performance correlation

<table>
<thead>
<tr>
<th>Variable</th>
<th>M</th>
<th>SD</th>
<th>1a</th>
<th>1b</th>
<th>2a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study (N = 9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1a. TMS internal</td>
<td>3.91</td>
<td>0.20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b TMS external</td>
<td>3.65</td>
<td>0.20</td>
<td>0.13*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a. Effectiveness</td>
<td>20.22</td>
<td>3.77</td>
<td>0.48*</td>
<td>-0.10*</td>
<td></td>
</tr>
<tr>
<td>2b. Efficiency</td>
<td>6.78</td>
<td>1.99</td>
<td>0.47*</td>
<td>-0.45*</td>
<td>0.66*</td>
</tr>
</tbody>
</table>

As we suspected, only team’s internal TMS score doesn’t explain team performance in large-scale distributed software development projects. From our correlation analysis we identify team R as a clear outlier when viewing only team’s internal TMS score. Team R have slightly bellow team’s internal TMS (TMSinternal = 3.74), but comparing to all other teams it was highly underperforming (Effectiveness = 11 out of 25, Efficiency = 3 out of 10). We argue, that Team R performance was influenced by other factors that may be relevant in large-scale software development. For further analysis, we also identified two other teams for comparison team A, that was somewhat underperforming (Effectiveness = 23, Efficiency = 8), given highest team’s internal TMS (TMSinternal = 4.33) and team C, from site B, that was highest performing team in that site (Effectiveness = 23, Efficiency = 6), while having team’s internal TMS (TMSinternal = 4.03). We also look at Team I (TMSinternal = 3.70) to see what practices may have hindered their TMS.

We further describe our findings about factors that previous research has identified that influence the development of TMS in teams (see Figure 1, four boxes on left). We also describe our findings about factors that previous research has put forward on the links between task characteristics and perceived team performance. And we also analyze and propose hypothesis regarding the impact of having access to differentiated knowledge from other teams and experts, and ability to use this expertise pool in a well-coordinated fashion on performance (see Figure 1, blue boxes). We analyze these factors in the context of the studied projects, team characteristics and networking needs. We describe
to what extent these factors may have a role in how teams can solve tasks in large-scale distributed software development projects. We use the data from the focus groups, interviews and the survey. Factors that can be quantified from our set of data (survey), we use in regression analysis (see Figure 1, blue boxes). Other factors we collect from the focus groups and the interviews using a pre-developed coding scheme (see Figure 1, white boxes on left and bottom). Those factors we weren’t able to quantify; thus we use them only for qualitative analysis. We use the questions identified by Lewis [1] as the basis for our coding scheme for the analysis of the focus group transcripts.

![Figure 1. Influences on TMS and performance](image)

**B. Project knowledge needs**

In this section we describe results from focus group analysis, to understand the context in which teams were operating as literature review shows that TMS have different implications on different types of teams and task types.

1) **System description and development methods**

Both systems we studied are very complex systems that contains several millions lines of code. In Company A that is sub-system that has multiple components and interfaces with other sub-systems. Due to the size and complexity of the system domain and product knowledge, becoming good requires years as it was described of one of developers from team A said: “problem is more in such big product and it's connected with other big products, you know, lots of different standardizations in radio network”. Company relies on Agile and Scrum development methodology that have been implemented for several years and have invested in building cross-functional teams that could implement a whole feature.

Development work is organized in seventeen self-managing cross-functional feature teams that comprise of members with different roles, where each team receives a high level description of a feature, and designs and implements the feature.

In Company B that is a complex system with multiple modules that have coupled dependencies and requires integration of knowledge about various areas and functionality. Company B puts large emphasis on product and process quality, and follows a V-model development methodology, in which tasks are structured and project members’ roles are task-specific. A good developer is said to have only a 50% overview of the system. However, as one of developers from team C said it takes multiple years to create necessary knowledge if no prior knowledge is available about the code: “it was very big challenge to master these pieces of code. And tracker database was enormous for us to handle. I think it took about five or six years to manage the code and describe the tracker cases, amount of tracker cases.” The role of each site is determined by the specialized expertise. The main Swedish location employs the majority of software developers, and thus software projects are usually led from this location with sub-project managers in each location. Hardware expertise resides in the second Swedish location, and thus hardware projects are led from there. There are a number of senior developers who have accumulated a lot of knowledge, who are informally known and
frequently contacted for consultation. Development work is organized in projects, which can include development of new generations of the system, new functionality development, roll-ups of large maintenance projects and pure maintenance projects.

2) Task descriptions and team knowledge needs

We have studied six Company A teams – three from the Swedish site (A, B and P) and three from the Chinese site of the company (G, N and R) and three Company B team - two from main Swedish site (C and I) and one from other Swedish site (S). The team profiles are presented in Table V. In case of Company B we also studied large group of developers in India. However, our pre-analysis showed that the Indian group is not formed as a team, but rather as a working group. Therefore, the team-level results from Indian group are not directly comparable and not included in the paper.

Of the selected 9 teams, 6 were cross-functional teams (including team A and team R), defined as teams that are constructed for a specific time-limited tasks, with members’ different specializations and areas [1], [37]; and 3 were functional teams (including team C), with members from the same department and area, consisting of team lead and comprising of developers working in the same area [1]. Most of the team members are engineers who worked on knowledge tasks - that is, tasks that depend on members having specialized knowledge and expertise [1]. From our sample, we don’t see difference in TMS between functional and cross-functional teams in terms of well-developed TMS and team performance. Both projects are complex and require apply large amount of differentiated knowledge that is pre-requisite of well-developed TMS. One exception may be Team I, where members specialize in certain areas without much collaboration between team members. This supports previous findings that while functional team members may posses specialized expertise, it may be less critical in order to perform well [24]. Type of each team is showed in Table IV.

3) Task descriptions

Tasks differed somewhat among the different types of teams. In Company A, a single work item assigned to a team is a feature. Feature is domain specific and differ by complexity and thus may require specific domain knowledge that team may or may not be possessed. However, over time a team can develop a specialization into a certain type of features that can be associated but is not limited to system functionality, systems layer or components. The most complex features require changes in several parts of the whole system. Team A in Company A is challenged with new and complex tasks frequently, as team member says: “Because features coming to us is more and more complex you cannot just split them in small, so.. I think that is challenge” and also receives challenges on tasks with higher interdependency: “As far as I know it is only Team A and (team not included in study) working cross subsystems in this way”. Those constant challenges may have helped team A to develop their TMS. Company A off-shore site in China in general receive tasks with less interdependencies and more simple. This was done as part of Company A strategy of distributed development. Team R has been working on what they perceive a big task for long time “For the kind of feature, we already worked for almost half year. It’s is big task”. That could have helped them to develop TMS within team, while still perceived as underperforming from managers.

In contrast, Company B practices detailed up-front design, in which only selected team members are involved, forming a virtual design team. However, when the development projects start, the work is usually well specified and coordinated through team

<table>
<thead>
<tr>
<th>Teams</th>
<th>Company experience (average)</th>
<th>Team experience (average)</th>
<th>Type of team</th>
<th>Team stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team C</td>
<td>11,3</td>
<td>6,1</td>
<td>Functional</td>
<td>Medium</td>
</tr>
<tr>
<td>Team S</td>
<td>17,2</td>
<td>9,1</td>
<td>Functional</td>
<td>Good</td>
</tr>
<tr>
<td>Team I</td>
<td>8</td>
<td>4,4</td>
<td>Functional</td>
<td>Medium</td>
</tr>
<tr>
<td>Team A</td>
<td>10,8</td>
<td>3</td>
<td>Cross-functional</td>
<td>Good</td>
</tr>
<tr>
<td>Team B</td>
<td>12,1</td>
<td>1,9</td>
<td>Cross-functional</td>
<td>Poor</td>
</tr>
<tr>
<td>Team P</td>
<td>17,5</td>
<td>2,4</td>
<td>Cross-functional</td>
<td>Medium</td>
</tr>
<tr>
<td>Team R</td>
<td>2,2</td>
<td>0,9</td>
<td>Cross-functional</td>
<td>Medium</td>
</tr>
<tr>
<td>Team G</td>
<td>2</td>
<td>0</td>
<td>Cross-functional</td>
<td>Medium</td>
</tr>
<tr>
<td>Team N</td>
<td>2,8</td>
<td>2,2</td>
<td>Cross-functional</td>
<td>Medium</td>
</tr>
</tbody>
</table>
leads, who further assign the tasks to individual team members. Each team specializes in a particular part of the system, a system component. Work complexity highly depends on the project lifecycle. New development requires upfront design and requires integration of multiple team members’ knowledge, as developer from team C mentions: “It's a very complex system. So you don't know everything about it.” Further development activities are less interactive and can be described as medium, but knowledge coordination is important also in the later stages as complex cases emerges due to systems complexity and technical and functional interdependencies. This was evident in team C responses: “That's difficult to say. Sometimes under qualified, sometimes overqualified.” Team I, however, often receives task that don’t have cooperative nature and then team leader will divide tasks to team members according to specializations, as he says: “Then I try to divide”, further, he says that that may be weakness of team: “Weakness in this system if one leaves or... we will lose at least speed on that area that you need to do correction on.” Specialization without need to for coordinate this specialized knowledge may have hindered development of TMS for this team. In contrast, Team C mentions that they put a lot of emphasis on discussions together, team members’ multiple times mentioned: “We discuss a lot of things, but everyday, of course we sit and code alone. We are more discussing design principles”. Team C also had build redundancy in knowledge intentionally and have vision to introduce newcomers in team, giving them bug fix cases “it's best way to really” to facilitate discussions and feedback in team. “That's good opportunity to discuss that case with other team members. That's very important to get feedbacks where to look in code and whom should I talk to and ask questions”. This strategy helps develop better understanding of each others knowledge and benefits TMS development. Task complexity and task interdependency is presented in Table V.

C. Experience of working together
Experience of working together is considered as one of the factors that could lead to positive TMS within team [48]. Team cohesion also is reported as one of the factors that helps develop TMS within team [48].

We learned that developer experience across locations differs. Generally, the Swedish teams in both projects are characterized by their team member long company experience (around 10 years on average). However, experience of working together varies. In case of Company B, affiliation with current team is 5 or more years, while in case of Company A, where to some extent shuffling around people strategy has been employed in past years, affiliation with current team is around 2 years. Due to this strategy, team B had poor stability in recent years. However, team A has kept stable core for past 2 years and few changes in team are mitigated by team size: “when you have so large team, you are not so vulnerable to sickness or vacation, longer vacation, that’s advantage”. Members in Chinese teams of Company A have less company experience, around 2 and half years on average. Two of the Chinese teams have had occasional member changes and have been stable for two years, while team R is the most immature team with less than a year’s joint work experience. Team C, with exception of one developer (who had previous experience working in this team), has kept core for several years. And while Team I has good team experience in average, they have three members (including team leader), with less then 2 years’ experience in team. Company experience of team members, experience in team, team stability are presented in Table IV.

D. Team networking needs
In both cases projects are large-scale distributed development that can be characterized as an environment in which nobody knows everything. Thus, teams varying on task complexity and task familiarity are required to have coordination on three levels: 1) among team-members, 2) between teams, and 3) between teams and the rest of the organization. The ability to maintain the knowledge needed and the ability to coordinate the work, explained by the theory of TMSs, are at the focus of our investigation.

1) Expertise
In Company A, some teams receive more unfamiliar features then others (see Table VI). In most cases
features are taken care of by one team, and represent a piece of functionality or a requirement requested by the system owner. Features can become complex then required to adhere certain standards or specific customer requirements. Implementing such complex features puts high demands on the team and the knowledge possessed by or accessible to the team. More complex features may require interaction with other teams and large number of formal and informal experts in company. Team A often needs to rely on experts as they are solving complex and unfamiliar tasks, but they also note that “it's also comfortable on relaying on experts in different areas”.

Some mature teams in Sweden are challenged by new type of tasks. Notably new teams in China receive more simple tasks. Besides new feature development, teams are involved in fixing bug reports. Also teams working towards a specific customer are often required to have a broader expertise to be able to complete any feature or bug fix in the sub-system.

In Company B, team members may be involved simultaneously in several projects. Regular development usually doesn’t require much interactions as teams specialized in certain areas, but teams are required to approve their solutions to safety engineers. Bug fixing is usually performed by the team responsible for corresponding part of the system. However, locating a bug in the system can be challenging. A complex trouble report may require a special task force consisting of experts from different teams, in order to locate the bug and fix it. This is mainly due to the system knowledge differentiated across multiple disciplined teams.

<table>
<thead>
<tr>
<th>Teams</th>
<th>Task complexity</th>
<th>Task interdependency</th>
<th>Task familiarity</th>
<th>Reliance on solving complex tasks</th>
<th>Reliance on solving familiar tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team C</td>
<td>Medium</td>
<td>Medium</td>
<td>Familiar</td>
<td>Networking</td>
<td>Networking</td>
</tr>
<tr>
<td>Team S</td>
<td>Medium</td>
<td>Medium</td>
<td>Familiar</td>
<td>Documentation</td>
<td>Document. /Skills</td>
</tr>
<tr>
<td>Team I</td>
<td>Medium</td>
<td>Medium</td>
<td>Familiar</td>
<td>Documentation</td>
<td>Skills</td>
</tr>
<tr>
<td>Team A</td>
<td>Complex</td>
<td>High</td>
<td>Unfamiliar</td>
<td>Networking</td>
<td>Networking</td>
</tr>
<tr>
<td>Team B</td>
<td>Varying</td>
<td>Medium</td>
<td>Varying</td>
<td>Networking</td>
<td>Networking</td>
</tr>
<tr>
<td>Team P</td>
<td>Varying</td>
<td>Medium</td>
<td>Familiar</td>
<td>Networking</td>
<td>Skills</td>
</tr>
<tr>
<td>Team R</td>
<td>Simple</td>
<td>Low</td>
<td>Familiar</td>
<td>Networking</td>
<td>Documentation</td>
</tr>
<tr>
<td>Team G</td>
<td>Simple</td>
<td>Low</td>
<td>Familiar</td>
<td>Networking</td>
<td>Networking</td>
</tr>
<tr>
<td>Team N</td>
<td>Simple</td>
<td>Low</td>
<td>Unfamiliar</td>
<td>Networking</td>
<td>Networking</td>
</tr>
</tbody>
</table>

2) Knowledge needs when solving tasks
Knowledge needs and need for networking differs between task familiarity. When solving unfamiliar and complex tasks, 7 out of 9 teams reported that they rely on networking, e.g. knowledge embedded within, available through, and utilized by interactions among individuals and their networks of interrelationship. Other two teams reported that they rely documentation, e.g. institutionalized knowledge and codified experience, often, for example, on source code. However, then solving more familiar and less complex tasks team members mostly reported a shift to some extent from networking to use of their own expertise and knowledge. However, even teams that rely on documentation had medium size network (total unique contacts of 20-30 reported in survey), which was further determined on self-reported knowledge network in focus groups. Before showing our preliminary results in feedback sessions to representatives from both companies, we asked how much contacts they estimate that a team would have. They responded that around 10 contacts in their networks. However, team A reported (both in survey and focus groups) larger network then comparing to all other teams with many unique contacts. Team R, in contrast, reported very small network, where most of interactions are with dedicated technical experts. Team members reported knowledge and status updates as important because they felt that otherwise developers and teams may loose the “whole picture”. Team R
members agreed that improvement on “is communication with other people outside the team” is important”. Team members acknowledged that many times a lack of overview of the system is the main reason for failure in delivery on time or quality. That was the reason why Company A has dedicated technical experts who has the role of consulting feature teams and approving feature solutions. Two Swedish teams (B and P) have dedicated technical experts working as part-timers in the team, while other teams have to coordinate their work with experts that are located outside of the team.

Table V. Team network size

<table>
<thead>
<tr>
<th>Team</th>
<th>Local</th>
<th>Remote</th>
<th>External</th>
<th>Unique external contacts per respondent</th>
<th>External connections per respondent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team C</td>
<td>15</td>
<td>8</td>
<td>23</td>
<td>5,8</td>
<td>8,3</td>
</tr>
<tr>
<td>Team S</td>
<td>6</td>
<td>16</td>
<td>28</td>
<td>5,6</td>
<td>5,6</td>
</tr>
<tr>
<td>Team A</td>
<td>14</td>
<td>5</td>
<td>19</td>
<td>3,8</td>
<td>5,8</td>
</tr>
<tr>
<td>Team A</td>
<td>36</td>
<td>12</td>
<td>48</td>
<td>8</td>
<td>11,5</td>
</tr>
<tr>
<td>Team B</td>
<td>24</td>
<td>2</td>
<td>26</td>
<td>5,2</td>
<td>7</td>
</tr>
<tr>
<td>Team P</td>
<td>22</td>
<td>3</td>
<td>25</td>
<td>5</td>
<td>6,2</td>
</tr>
<tr>
<td>Team R</td>
<td>9</td>
<td>2</td>
<td>11</td>
<td>1,8</td>
<td>4,2</td>
</tr>
<tr>
<td>Team G</td>
<td>17</td>
<td>4</td>
<td>21</td>
<td>2,6</td>
<td>5,25</td>
</tr>
<tr>
<td>Team N</td>
<td>22</td>
<td>7</td>
<td>29</td>
<td>5,8</td>
<td>8</td>
</tr>
</tbody>
</table>

In contrast, Company B doesn’t have dedicated technical experts in Sweden who approve solutions like in Company A, but some team members are recognized for their experience and seniority as formal or informal experts in a particular part of the system. Team C members are very social and a lot of tasks are handled through discussions and knowledge sharing among the team members: “there are no closed doors”. Because they rotate the tasks in different parts of the system and do not follow strict specialization, knowledge sharing is often required. Communication with team C members where most often mentioned by others. Detailed overview of team network size and unique contacts is presented in Table VI.

E. Factors that impact TMS development – Regression analysis results

Our results suggest that only team’s internal TMS is not sufficient to explain performance. We used our results from focus group analysis to conclude that teams often need to rely on outside knowledge to solve tasks. However, as communication with other teams and experts takes time and effort, efficiency of such a network is important. Previous research on TMS shows that successful expertise retrieval is more likely to happen when team members have higher awareness of expertise distribution and larger pool of accessible experts in their social network [44].

Thus, we further investigated two factors:
1. team’s external TMS, as that may facilitate locating experts in the network to solve tasks in a timely manner and
2. number of non-overlapping contacts outside the team, as that may indicate whether the team uses its network efficiently, e.g. if team members have a lot of unique contacts in the network, there is less communication overhead, as team members can spend more time to communicate with different experts.

Table VI. Regression analysis

<table>
<thead>
<tr>
<th>Regression Statistics</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple R</td>
<td>0,50</td>
<td>0,84</td>
<td>0,93</td>
</tr>
<tr>
<td>R Square</td>
<td>0,25</td>
<td>0,71</td>
<td>0,86</td>
</tr>
<tr>
<td>Adjusted R Square</td>
<td>0,00</td>
<td>0,61</td>
<td>0,77</td>
</tr>
<tr>
<td>Standard Error</td>
<td>3,76</td>
<td>2,35</td>
<td>1,80</td>
</tr>
<tr>
<td>Observations</td>
<td></td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>

First, we examined the relative effects of team’s internal TMS and other control variables, namely team’s external TMS and the number of unique contacts per team member, on overall performance using a standard multiple regression analysis. We created two models - that team’s internal TMS score and team’s external TMS score will positively predict perceived effectiveness of team by external evolution (H1) and that team’s internal TMS score number of unique contacts team have will have positive influence on perceived effectiveness of team by external evolution (H2). The traditional factors model had an adjusted R2 of 0,00 for H1 and R2 of 0,61 for H2 (Significant F < 0,01, all
variable levels under \( p < .05 \). Thus both of these hypotheses were not supported.

Further, we created a model that all three mentioned variables together will predict team performance. Thus, our hypothesis is that in large-scale software development context teams with high score of combination of team’s internal TMS, team’s external TMS and number of unique contacts team have will have positive influence on perceived effectiveness of team by external evolution (H3).

Table IV provides the results of the regression analysis. The traditional factors model had an adjusted \( R^2 \) of 0.77 (Significant \( F < 0.01 \), all variable levels under \( p < .05 \)) and thus was a significant predictor of team effectiveness. Thus, our hypothesis that team’s external TMS and team’s member unique contacts per member are associated with team performance, is supported.

V. DISCUSSION

We have described the knowledge network characteristics and team knowledge needs in two large software companies in Sweden, Company A and Company B. We present partial replication of studies of measuring TMS [24] measuring expertise coordination and measuring team performance [2] measuring TMS and knowledge networks [55] to examine large-scale software project setups and their effect on TMSs and team knowledge needs in large-scale distributed development.

Previous research shows that organizational aspects of social networks and knowledge networking in particular in software development has been undervalued for a long time [6]. This was supported by our finding that while managers generally where aware of need of communication between teams and supporting roles, they also greatly underestimated (more than 50\%) amount of actual networking efforts and, as a consequence, the time and resources required to support networking. Teams in Company A reported that meeting space is not sufficient or not everybody is aware of the available communities of practices (CoP), while CoP culture was not formalized in Company B at all. This also contrasts our finding that when solving complex or unfamiliar tasks most development teams in Company A and half of the teams in Company B perceived to rely primarily on teamwork and networking with contacts in their networks.

A. TMS and performance

Addressing our first research question, we have identified that generally well-developed TMSs in large-scale distributed development correlates with performance, as promised by studies from other knowledge-intensive field settings. However, we learned that while there is a positive relationship there are multiple additional factors that may have an influence.

We investigated few of those factors which we perceived as more important and identified that due to complexity of large-scale distributed development, teams need well-developed TMSs on an organizational level with other teams and supporting roles, to be perceived as well-performing. This is understandable, since we also learned that a single team cannot possess all the necessary knowledge within the team, usually leading to some specialization in certain area or specialization in module and using supporting role expertise to gain the system knowledge on the higher level. Furthermore, this knowledge network of teams and experts should be efficient as it requires time to build and use connections with other project members and team member should have some level of specialization not only in a product knowledge area, but also in sharing networking burden.

B. Relevance of findings for industry

Based on findings of our research and previous research from different disciplines [62], we argue that if an organization is aware of its existing TMS network, it can make decisions about how they use and develop further this network in the most efficient way and adjust other project management practices to support development of teams’ TMSs. Our findings suggest that in large-scale software development projects there is link (affect) between the team formation and task allocation practices, and the development of teams’ TMS. Our approach on how to examine knowledge networks in large-scale software development can be used by both practitioners and researchers as a cause and effect tool for understanding the status of TMS of their
teams and using knowledge network data to improve coordination and collaboration between teams and supporting roles of large-scale software development.

C. Practices that support TMS

Comparing our focus group analysis with TMS scores, we identified the characteristics of teams with well-developed TMS and teams with weakly developed TMS and analyzed practices that may have been the reason of that. From the use of previous studies of TMS, we identified that it is important to understand what tasks a team receives [56]. As previous research [44] and our findings suggest, task characteristics play role on both development of TMS and benefitting from well-developed TMS. Based on our findings, we learned that in large-scale software development teams receive tasks all types of task processes - “produce”, “choose” and “execute” [56]. Tasks involve idea-generation and require generating multiple ways on how to solve problems. It also involves group decision-making. Large-scale context especially requires group knowledge that is diverse and specialized to cope with environment where “nobody can know everything” about the project. Team members are requested to collaborate and integrate their knowledge from all available resources to solve tasks. This suggests that TMS, in other words efficient knowledge coordination, is particularly relevant in large-scale software development.

In our study, we found that both cross-functional (Company A) and functional teams (Company B) teams demonstrated ability to develop good TMS. And this is in line with previous research [24]. We found teams from both organizations that received more cooperative tasks with more task interdependency and cooperative nature showed better TMS scores comparing to teams that received tasks that could be divided and executed alone and were assigned by the team leader (in one case). Team stability also affected TMS, were teams with better stability tended to have better TMS. Teams that had better team cohesion, discussions about solutions, giving feedback and relying on each other also showed better developed TMS and better performance scores.

We suggest that managers of software development companies and software teams should be aware of practices that may help or hinder development of TMS, both, within teams and with other teams and supporting roles.

D. Implications for managers and for developers working in teams

There are several practices from our results that is supported by previous research that would be applicable in software engineering for managers of large-scale projects. These practices are attributed to training teams, assembling teams, task characteristics and communication.

Previous research shows that teams that have longer experience working together [50] request and accepted backup from one another more. Our results show that in both organizations teams that had longer experience working together had better developed TMS. We therefore, suggest that keeping team together for longer time will help those teams develop TMS and have better team performance on long run. Previous research on TMS suggest that groups that train together develop better TMS [37]. From our results we suggest that managers can support team’s internal TMS development by challenging team with new and more difficult tasks, as it was especially seen in case of Team A.

Further, our findings suggest that task allocation strategy for teams should take account need for development of team’s TMS and need to benefit from well-developed team’s TMS. To develop team’s TMS, managers should allocate tasks so that teams receive cooperative and divisible tasks. These tasks will more likely motivate team members to learn what other team members know and further develop ways of strengthening the specialization in the team [56].

Previous research of TMS also shows that face-to-face communication have a large role in developing TMS [61]. Allocating part-time team members will also have a negative effect on the development of TMS. Allocating team members to multiple projects our task-force teams that requires spending time away from team will also have similar negative effect. For team members should have practices that facilitate face-to-face communication such as daily stand-up meetings will help develop team’s internal TMS.
Our results from regression analysis that team’s external TMS and the number of unique contacts in the team will have positive influence on perceived effectiveness of the team suggests that both managers and team members should be aware and support external networking processes. These results were further supported by focus group results that most of teams rely on networking, when are required to solve complex or unfamiliar tasks. This implicates that managers should work on creating a network of available experts and supporting roles for team members to gain the necessary knowledge to solve their tasks. Managers should acknowledge and support need to spend more time to networking when allocating complex and/or unfamiliar tasks. For managers this implicate that they should support team communication needs with allocating time to participate in CoPs and creating enough work-space for meetings within and between teams. For team members that would mean that dividing who will attend and attending CoP and other similar meetings will help develop team’s external TMS.

VI. CONCLUSION

A. LIMITATIONS AND FUTURE WORK

A number of limitations inherent in the study should be recognized. The first is that the empirical data related to the social network are self-reported. In other words, the study investigated the perception of the network and did not attempt to calculate or determine an objective measurement of the social network. Consequently, knowledge and information sharing could have been exaggerated or underestimated by the responses limiting reliability of the conclusions.

The second, generalizability of study findings is an issue because our sample empirical data was collected from a limited number of teams and within two projects and thus may not be sufficiently representative of teams developing software at other organizations. Our observations might be influenced by particular and distinctive characteristics of the projects, such as culture, development methodology and project characteristics. Other studies, that will gather empirical data within similar conditions would be necessary in order to dispelling this threat to external validity.

The third, limitation of reliability is the use of subjective rather than objective performance measures. In the future, researchers may want to collect more objective measures of performance. However, that has been proven as difficult in the software development context.

The fourth, limitation of generalizability is that complex and real work environment limits our ability to control variables that may influence team performance in such settings. Further studies about complexity of this environment is needed to eliminate the threat to external validity.

The fifth limitation is that real networks are not static, but rather dynamic. Relationships between people may change and communication networks evolve over time. In the current study we examined only a snapshot of the networks. Future research should compare transactive memory systems over time and observe their evolution.

The sixth limitation is that reliability analysis for the latent variable model of TMS processes, we recognize that there might be threats to construct validity. Further effort to re-evaluate, validate and potentially improve the questions used for the operationalization of the model is needed.

B. CONCLUSIONS AND FUTURE WORK

Our main conclusions of our study are threefold. First, we conclude that lessons learned from TMS research can be applied in large-scale distributed development. We conclude that here is link (affect) between the team formation and task allocation practices, and the development of teams’ TMS. This suggest future need for a longitudinal study when where we can measure the development of teams’ TMS based on the different organization practices.

Second, we conclude that link between well-developed team’s internal TMS and team performance was mediated by team’s external TMS and efficiency of team’s external TMS. However, we conclude that other factors that our study was not able to quantify may have role on team performance as well, thus future studies to investigate role of those factors is needed.

Third, measuring TMS in large-scale work setting is challenging. This is due to reason that large-scale context requires capturing very complex environment. To fully understand large-scale context, we need to capture TMS in two levels –
team and organizational. To further understand of TMS in organizational level we would need to capture actual knowledge flows that are related to software development and work coordination.

For the future work, research on networks of different knowledge flows would yield additional understanding of how different knowledge is shared within organization. Comparing those networks with formal structures of organization would help identify weak points in organization communication structure. Research on type of networks in large-scale organization could yield implications on networking practices as distribution of contacts seems be more in form of “rich get richer”, i.e. few project members that have good communication skills and already large set of contacts tend to accumulate large part of network connections.

REFERENCES


19


## APPENDIX I

<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) Q1. I am aware of this person’s area of expertise</td>
<td>(Q3) Q2. I am aware of the knowledge that this person needs for his/her job</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessibility Credibility</td>
<td></td>
<td>(Q2) Q4. It is easy for me to access this person (Q9/Q4) Q5. The information I receive from this person is important for my job</td>
<td></td>
</tr>
<tr>
<td>Communication frequency</td>
<td>(Q4) Q3. How often do you transfer knowledge to this person? (Q5. What kind of knowledge do you transfer to that person?)</td>
<td>(Q6. How often do you receive knowledge from that person?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Q6. How often do you request knowledge from this person? (Q7. What kind of knowledge do you receive from that person?)</td>
<td></td>
</tr>
</tbody>
</table>
**APPENDIX II - QUESTIONNAIRE**

*note – question sequence was a bit different between both projects. In Company A questionnaire was as follows, in Company B questionnaire question 8 was question 4.

**note – question 8 was omitted in data analysis, as it was asked only in Company A and Company B offshore site.*

<table>
<thead>
<tr>
<th></th>
<th>Question</th>
<th>Scale/Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>I exchange knowledge with [Name Surname]: *</td>
<td>Free recall</td>
</tr>
<tr>
<td>2. (Q1)</td>
<td>I am aware of that person’s area of expertise</td>
<td>5-Point Likert scale (Strongly disagree – Strongly agree)</td>
</tr>
<tr>
<td>3. (Q2)</td>
<td>It is easy for me to access that person</td>
<td>5-Point Likert scale (Strongly disagree – Strongly agree)</td>
</tr>
<tr>
<td>4. (Q3)</td>
<td>I am aware of the knowledge that person needs for his/her job</td>
<td>5-Point Likert scale (Strongly disagree – Strongly agree)</td>
</tr>
<tr>
<td>5. (Q4)</td>
<td>How often do you transfer knowledge to that person?</td>
<td>5-Point Likert scale (Never – Daily)</td>
</tr>
<tr>
<td>6. (Q5)</td>
<td>What kind of knowledge do you transfer to that person?</td>
<td>Free recall</td>
</tr>
<tr>
<td>7. (Q6)</td>
<td>How often do you receive knowledge from that person?</td>
<td>5-Point Likert scale (Never – Daily)</td>
</tr>
<tr>
<td>8. (Q7)</td>
<td>What kind of knowledge do you receive from that person?</td>
<td>Free recall</td>
</tr>
<tr>
<td>9. (Q8)</td>
<td>I trust that the knowledge I receive is credible</td>
<td>5-Point Likert scale (Strongly disagree – Strongly agree)</td>
</tr>
<tr>
<td>10. (Q9)</td>
<td>The information I receive from the person is important for my job</td>
<td>5-Point Likert scale (Strongly disagree – Strongly agree)</td>
</tr>
</tbody>
</table>
9  APPENDIX III – SURVEY INSTRUCTIONS

You are requested to identify persons who you exchange knowledge with (receiving from or transferring to).
Knowledge includes anything related to your work (knowledge of a product area or functionality, source code characteristics, technical knowledge, process descriptions, administrative guidelines).
You can enter information about as many persons as you wish. These should represent your knowledge network.
You can revisit and edit your responses before your link expires [date].
0. Introduction

The purpose of this research study is to understand the links between team performance and the use of human, social and organizational capital to accomplish a task. This will be achieved by performing the following steps:

1. We first aim at understanding the complexity and nature of work being performed, and the amount of knowledge required to perform the work.
2. We then aim at evaluating the presence of the necessary knowledge in the team, and how this knowledge is developed and coordinated among the team members.
3. If the team does not possess the needed knowledge, we aim at understanding whether this knowledge is acquired from other people in the organization, or the institutionalized knowledge.
4. The results will be compared with the team performance evaluation (by the team, and by the stakeholders).

Based on the results we should be able to diagnose the reasons for gaps in performance (if any) and suggest potential improvements areas with respect to human, social or organizational capital, where social capital comprises teamwork and knowledge network outside of the team.

1. Round-Robin 15 min

Ask everyone to introduce him or herself and describe their professional experience:
- Number of years of experience in software development proxy for technical knowledge
- Number of years of experience in the company proxy for domain knowledge
- Number of years of experience in the project proxy for product knowledge
- Number of years of experience in the team proxy for team stability
- Percentage of working time dedicated to the team proxy for accessibility

2. Project context and timeline 10 min

Identify evolution phases on the timeline and agree on the scope for the further steps.
- Ask everyone to think of any major events during the project evolution (e.g changes in the team, important releases, architectural, technical, organizational, or task-related changes). Discuss and add to the timeline.
- When did the team start their involvement?

3. Detailed task information 20 min

Ask the team about their work tasks.
- What do you see as a task? What type of tasks do you receive?
- How are the tasks received?
- Did the tasks change over time?
- How complex are the tasks?
- What type of knowledge is required to perform the task?

4. Human capital – Presence of expertise in the team 20 min

Ask to evaluate the percentage of the necessary knowledge in the team:
- Technical expertise – knowledge about the programming languages, databases and development tools
- Domain expertise – knowledge about the application domain area and client (or user) operations
- Product expertise – the knowledge of the source code, software design and architecture

Specialization – ask the team members about the amount of specialized and shared knowledge.
- Do team members have any specialized knowledge that no other team members have?
- Do different team members take responsibility for expertise in different areas?
- Is the specialized knowledge of several different team members needed to complete tasks?
• Does the team have a good “map” of each others’ talents and skills?

**Pause 10 min**

5. **Social capital – Teamwork and knowledge coordination in the team 40 min**

Short retrospective – what works well and what does not work well or could be improved in the team?

**Coordination in the team** – Ask the team members to evaluate their teamwork skills:

- Is the team good at solving problems together?
- Are decisions made together?
- Are team members good at shifting their workload?
- Do team members share common goals?
- Do team members give each other feedback (positive and negative)?
- Do you exchange/share the knowledge about how to perform the team task?
- Do more knowledgeable team members provide others with hard-to-find or specialized knowledge?
- Do team members know what others are doing?
- Does the team learn from experience?
- Does the team have misunderstandings about what to do?
- Does the team need to backtrack and start over a lot?
- How are the new team members being introduced?

**Credibility** – ask the team whether they rely on the expertise of their team members:

- Are you comfortable accepting procedural suggestions from each other?
- Do you trust that other members’ knowledge about the project is credible?
- Are you confident relying on the information that others bring to the discussion?
- Do you have a tendency to double-check the information received?

6. **Accessibility of expertise outside of the team 30 min**

**Coordination with those outside of the team** – Ask each team member to identify their external interactions (on a set of post-it notes). Distinguish three types of interactions. Draw the team interactions on the whiteboard.

**Knowledge retrieval:**

- Who provides the knowledge or expertise needed for the team’s tasks?
- How often?
- Is the knowledge received from each of the identified people/communities credible?

**Administrative coordination:**

- How are schedules, policies, and documents managed?
- Are there any regularly scheduled team meetings with any stakeholders to discuss the progress?
- Are requirements/design review meetings being organized?
- Are design inspections being organized?
- Is the knowledge received from each of the identified people/communities credible?

**Knowledge sharing:**

- Who does the team share expertise with?
- How often?

7. **Organizational capital 20 min**

Discuss whether the team relies on any of the organizational assets in their work:

- Software source code
- Software architecture
- Software documentation
- Process documentation, formal policies and procedures
- Organizational culture and ways of working
- Development environment and tools
- Knowledge-based infrastructure

8. Reliance 15 min

Ask participants to reflect on the balance-proportion (from 100%) of the skills, teamwork, interaction outside of the team and product capital in each phase or for every task. What is the strongest IC component and why?

9. Performance 10 min

Ask participants to evaluate the perceived performance of the team in each phase. How easy it is/was to perform the tasks in each phase on the following scale:
1. The tasks are very easy to handle,
2. The tasks are handled without any major problems,
3. The tasks require some effort. Occasionally, major issues may occur. In most cases, it works quite smoothly,
4. The developers have a hard time handling their tasks. Major problems occur more often than not,
5. It is almost impossible to handle the tasks, and they take a long time.
11 **APPENDIX V – TEAM PERFORMANCE EVOLUTION QUESTIONS**

To have a little more objective information about the teams, we would like to collect line manager's view on the team performance. Could you, please, give your assessment of the Team X, using the following scale.

---

(the scale assumes that 1 is lowest and 5 is the highest grade)

---

Effectiveness – How well the project team performs compared to other software teams:

- Work quality: (1 – 2 – 3 – 4 – 5)
- Team operations: (1 – 2 – 3 – 4 – 5)
- Ability to meet project goals: (1 – 2 – 3 – 4 – 5)
- Extent of meeting design objectives: (1 – 2 – 3 – 4 – 5)
- Reputation of work excellence: (1 – 2 – 3 – 4 – 5)

Efficiency – How well the project team performs compared to other software teams:

- Adherence to schedules: (1 – 2 – 3 – 4 – 5)
- Adherence to budgets: (1 – 2 – 3 – 4 – 5)

---

The goal is to relate each team to other teams when comparing their performance. Work quality is about the number of bugs found in the code. Team operations means how the team operates as a team (teamwork). Ability to meet project goals includes understanding of the requirements and different constraints. Extent of meeting design objectives can be related to the satisfaction with their work by the TARs. Reputation of work excellence is how they are perceived from outside of the team, how knowledgable, how good at what they do, whether other teams want to be like them. It is a bit subjective measure.
Table I: Results of SLR for TMS benefits

<table>
<thead>
<tr>
<th>Index</th>
<th>Research method</th>
<th>Setting</th>
<th>Institute</th>
<th>Team Size (n)</th>
<th>Research method</th>
<th>Setting</th>
<th>Institute</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS1</td>
<td>F-S</td>
<td>Field</td>
<td>69 teams</td>
<td>3+</td>
<td>d</td>
<td>2</td>
<td>d in 49 individuals</td>
</tr>
<tr>
<td>SS2</td>
<td>L-M</td>
<td>Laboratory</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td>69 teams</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benefits or Research Result</th>
<th>Context</th>
<th>Team Size</th>
<th>TMS measures</th>
<th>Group Size</th>
<th>Settings</th>
<th>Research method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team stability, team member familiarity, and interpersonal trust had a positive impact on the transactive memory system and also had a positive influence on team learning, speed-to-market, and new product success. The impact was higher when there was a higher task complexity.</td>
<td>Performance Evaluation</td>
<td>10-20 teams</td>
<td>d - latent variable model proposed by Lewis (2003), i - modified latent variable model, other</td>
<td>2</td>
<td>Laboratory, dyad pairs of teams</td>
<td>L-M</td>
</tr>
<tr>
<td>The results indicate that both nonverbal and paralinguistic communication play an important role in the retrieval of knowledge in transactive memory systems.</td>
<td>Knowledge domains</td>
<td>1</td>
<td>d - error rate in task</td>
<td>2</td>
<td>Laboratory, dyad pairs of teams</td>
<td>L-M</td>
</tr>
</tbody>
</table>

Table I: Results of SLR for TMS benefits

Context: Knowledge domain in which data was collected:

- TMS measures: d - direct, i - latent variable model proposed by Lewis (2003), i - modified latent variable model, other
- Setting: number of participants, either individual, dyad pairs of teams, or dyad groups
- Research method: F - Field study, L - Laboratory experiment

Following Table I, this systematic literature review results that was done as part of Research Methodology course and further extended with more in-depth analysis.
Groups that experience partial membership changes tend to rely on the TMS that experienced individuals developed in their original group, and this change so is

Study demonstrates that TMS influence group learning and learning transfers.

The findings encourage a reexamination of certain assumptions of TMS theory.

The TM directories are proposed to have positive influence with service capital.

Experience vacuously. We
decide that transactive memory systems have evolved, and that changes within groups alters the decision-making capability of group leaders on dual decisions.

The different distribution of information inherent in transactive memory systems provide the best basis for conducting group-based decision-making.

Team performance efficiency

Product development

was found to hold for task and external evaluations. The positive relationship with group performance, and potentially internal group performance, and

positively related to group goal performance, external group evaluations, and

positively related to group goal performance, external group evaluations,
Teams with initially distributed expertise and familiar members are more likely to develop a TMS. Frequent face-to-face communication also led to TMS emergence, but communication via other means had no effect. Teams with more established TMSs later benefited from face-to-face communication, but they were less helped by frequent communication via other means, suggesting that transactive retrieval processes may have been triggered during face-to-face communication and suppressed during other forms of communication. TMSs were positively related to team viability and team performance, suggesting that developing a TMS is critical to the effectiveness of knowledge-worker teams.
<table>
<thead>
<tr>
<th>Reference</th>
<th>Sample Size</th>
<th>Design</th>
<th>Measure</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS17</td>
<td>3-9 teams</td>
<td>Survey</td>
<td>117</td>
<td>Consistency</td>
</tr>
<tr>
<td></td>
<td>196 individuals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS18</td>
<td>5-6 dyads</td>
<td>Survey</td>
<td>40</td>
<td>Error rate</td>
</tr>
<tr>
<td></td>
<td>80 individuals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS19</td>
<td>5-6 dyads</td>
<td>Questionnaire</td>
<td>118</td>
<td>Knowledge organization</td>
</tr>
<tr>
<td></td>
<td>216 individuals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS20</td>
<td>Single team</td>
<td>Survey</td>
<td>43</td>
<td>dyadic-survey</td>
</tr>
<tr>
<td></td>
<td>43 individuals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS21</td>
<td>18 teams</td>
<td>Survey</td>
<td>218</td>
<td>All levels</td>
</tr>
<tr>
<td></td>
<td>556 individuals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS22</td>
<td>3-9 teams</td>
<td>Survey</td>
<td>3-9</td>
<td>dyadic-survey</td>
</tr>
<tr>
<td></td>
<td>56 individuals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS23</td>
<td>1 team</td>
<td>Survey</td>
<td>43</td>
<td>dyadic-survey</td>
</tr>
<tr>
<td></td>
<td>43 individuals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Article</td>
<td>Reference list</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>