Information and Software Technology 86 (2017) 71–86

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Software teams and their knowledge networks in large-scale software development

Darja Šmite[a,*], Nils Brede Moe[a,b], Aivars Šāblis[a], Claes Wohlin[a]

[a] Blekinge Institute of Technology, Karlskrona, Sweden
[b] SINTEF ICT, Trondheim, Norway

ARTICLE INFO

Article history:
Received 4 May 2016
Revised 10 January 2017
Accepted 11 January 2017
Available online 19 January 2017

Keywords:
Large-scale software development
Knowledge networks
Social capital
Intellectual capital
Teams
Agile
Cross-functional
Feature teams
Empirical
Case study

ABSTRACT

Context: Large software development projects involve multiple interconnected teams, often spread around the world, developing complex products for a growing number of customers and users. Succeeding with large-scale software development requires access to an enormous amount of knowledge and skills. Since neither individuals nor teams can possibly possess all the needed expertise, the resource availability in a team's knowledge network, also known as social capital, and effective knowledge coordination become paramount.

Objective: In this paper, we explore the role of social capital in terms of knowledge networks and networking behavior in large-scale software development projects.

Method: We conducted a multi-case study in two organizations, Ericsson and ABB, with software development teams as embedded units of analysis. We organized focus groups with ten software teams and surveyed 61 members from these teams to characterize and visualize the teams' knowledge networks. To complement the team perspective, we conducted individual interviews with representatives of supporting and coordination roles. Based on survey data, data obtained from focus groups, and individual interviews, we compared the different network characteristics and mechanisms that support knowledge networks. We used social network analysis to construct the team networks, thematic coding to identify network characteristics and context factors, and tabular summaries to identify the trends.

Results: Our findings indicate that social capital and networking are essential for both novice and mature teams when solving complex, unfamiliar, or interdependent tasks. Network size and networking behavior depend on company experience, employee turnover, team culture, need for networking, and organizational support. A number of mechanisms can support the development of knowledge networks and social capital, for example, introduction of formal technical experts, facilitation of communities of practice and adequate communication infrastructure.

Conclusions: Our study emphasizes the importance of social capital and knowledge networks. Therefore, we suggest that, along with investments into training programs, software companies should also cultivate a networking culture to strengthen their social capital, a known driver of better performance.

1. Introduction

Nowadays, large-scale software development projects are characterized by unprecedented scale in terms of lines of code, amount of data stored, accessed, manipulated, and refined, as well as the number of connections and interdependencies, hardware and computational elements, customers and users, and, of course, the number of developers involved in the projects. Furthermore, today's projects are technologically complex from both innovation and design perspectives, and developers and leaders have a limited ability to learn from previous efforts because all large-scale projects are unique [1].

Large-scale projects pose a great risk and are often associated with cost overruns, late completions, and outright project failures [2,3]. One reason for the high risk is the complexity of large-scale project governance structures, which is often proportional to the number of development teams involved. Delivering results frequently and iteratively requires work and knowledge coordination on different levels, e.g., the portfolio, project, and team levels. Additional supporting roles (e.g., portfolio management) are critical in large-scale projects for managing the exponential growth of interdependencies and mitigating associated risks [4]. Furthermore,

* Corresponding author at: Blekinge Institute of Technology, Campus Gräsvik, SE-371 79, Karlskrona, Sweden.
E-mail address: Darja.Smite@bth.se (D. Šmite).

http://dx.doi.org/10.1016/j.infsof.2017.01.003

teams in evolving product development, which is often large in scale, require access to an enormous amount of knowledge and skills [5]. The expertise needed on the team level includes not only technical skills (programming languages and methodologies), but also teamwork and process knowledge, domain knowledge, and product knowledge, e.g., the architecture, source code structure, and allocation of concepts within the code.

The complexity and scale introduce three fundamental challenges. First, large-scale development reaches the point where hardly anyone knows everything about a system's development and evolution. Second, retaining original developers for decades is problematic because of leaves of absence, employee turnover, and retirements. Finally, large-scale projects often require the formation of new teams and the addition of new developers. The key question is then: how can software organizations efficiently cultivate the knowledge and skills needed in the development teams?

Evidently, enabling effective knowledge networks is crucial for succeeding in large-scale software projects. The need for such networks outlines the importance of social capital—a contextual complement to human capital (individual knowledge and skills)—which is an emerging concept in business, political science, and sociology [6]. Social capital refers to the actual and potential resources embedded within, available through, and derived from the network of relationships possessed by an individual or a social unit [7].

The increase in demand for knowledge-based economic activity, particularly product innovation, requires that we have a greater understanding of the factors that enable knowledge creation and sharing in a software team [8]. Moreover, because large organizations often have large and distributed multi-team projects, we must also understand what enables knowledge creation and sharing between teams, and between teams and the rest of the organization. Motivated by the importance of networking and social capital in large-scale software projects, we explore the actual knowledge networks and social practices of two large-scale projects in two software-intensive companies, Ericsson and ABB. Our research is exploratory in nature and is driven by the following research question:

RQ: What influences team knowledge networks and networking behavior in large-scale projects?

The remainder of the paper is organized as follows. Section 2 outlines related work. In Section 3, we describe our research methodology. In Section 4, we present our findings from the cases and cross-case analysis, which are further discussed in Section 5. Finally, Section 6 concludes the paper with a summary of major findings.

2. Background and related work

In this section, we describe related research on coordination in large-scale projects and the role of social capital in addressing coordination problems and knowledge needs.

2.1. Coordination in large-scale projects

Coordination is defined as the "management of interdependencies between activities" [9] and coordination mechanisms are the organizational arrangements that allow individuals to act collectively [10]. Interdependencies include shared resource dependencies and activity synchronization.

Software development is creative work, which means that a single optimal solution may not exist, and progress towards completion can be difficult to estimate [11]. One reason for this is that interdependencies between different pieces of work may be unknown or challenging to identify, making it difficult to know who should be involved in the work, and whether there is a correct order in which parties should complete their own specialized work [10].

Coordination in large-scale software development is of paramount importance since the work is carried out simultaneously by many developers and development teams [12]. In such projects, interdependencies are more uncertain than in small projects; therefore, teams need to know who the experts are and which experts to contact, particularly when they are outside the team or even at a different site. Licorish and MacDonell [13] studied global software teams and found that the availability of experts in teams' networks was linked to project-level performance.

Coordination can be either predefined or situated [14]. Predefined coordination takes place prior to the situation being coordinated. It typically consists of establishing written or unwritten rules, routines, procedures, roles, and schedules. Situated coordination, on the other hand, occurs when a situation is unknown and/or unanticipated. Those involved in the situation do not know in advance how they should contribute. They lack knowledge of what to achieve, who does what, how the work should be divided, in what sequence sub-activities should be done, and when to act. Consequently, in situated coordination, those involved must improvise and coordinate their efforts in an ad hoc manner.

Software development projects, particularly large-scale efforts, have a mix of predefined and situated coordination. Involved teams may, for example, already know the goal and the working process of the team, but they may not know who performs what outside the team, or they may know who does what but not when it is done. To compensate for a lack of predefined knowledge of how the activities will actually occur, teams and team-members must keep themselves updated on the status of the activity/situation to better understand who does what. Improving the knowledge transactions between team members and between teams by, for example, co-location and meetings, can strengthen awareness of the processes and activities.

Naturally, when team members are in close proximity to one another, particularly when they are in the same room, they become more aware of other members' work by seeing and overhearing their activities [15]. In addition, during, for example, daily meetings, team members constantly update each other on the status and become aware of the changes. However, putting every member of a large-scale project into the same room or the same meeting is often problematic. Empirical cases, e.g., Boden et al. [16], demonstrate the limitations of, for example, groupware tools, in supporting knowledge management, if not grounded in the work practices of the communities they aim to support. Similarly, Paasivaara and Lassenius [17] describe a large-scale development initiative at Ericsson, in which basic coordination mechanisms failed to support the coordination of 40 teams in three sites. Instead, knowledge sharing and work coordination in the Ericsson study was enabled through a number of communities of practice, which are known as groups in the organization that on a regular basis share a concern, or a set of problems [18]. Based on these findings, we propose that socialization-oriented coordination mechanisms are paramount in large-scale multi-team and multi-site environments.

2.2. Coordination by architecture

Coordination by architecture is commonly used to minimize the need to coordinate between teams and across geographic, cultural, and language boundaries. Ovaska et al. [19] found that participants in a large-scale distributed project coordinated their development work through software module interfaces. Relying on this strategy, each software modules could be developed separately, and thus the coordination problems could be mitigated.

However, experience shows that the effectiveness of coordination by architecture is limited, since system modules are never truly independent. Development of technically interdepen-

dent modules in isolation may lead to discrepancies, which often remain hidden until integration [20]. Additionally, organizations that follow modularized development and assign the work on a specific module to a single team, cultivate deeper specialized knowledge about that module, rather than a broader knowledge about different parts of the system. This potentially leads to more knowledge dependencies, if the modules have functional dependencies or complex integration points. Herbsleb et al. [20]. demonstrated that modularization alone is insufficient to overcome the challenges, and architecture-based, plan-based, and process-based coordination without the support of situated coordination is likely to fail.

One alternative to organizing the teams by modules in large-scale software development is assigning team responsibility for features. Coordination by features was explored by Paasivaara et al. [21] in a case study of large-scale globally distributed software development project using Scrum. A feature in a large-scale project can cover several sub-systems and modules. Paasivaara and colleagues found that the feature-based Scrum of Scrum meeting can be a good coordination mechanism, because a small group of people with common interests and goals could share, discuss and even solve problems together.

2.3. Coordination by networking

Given the premise that it is impossible for a single individual to possess all the knowledge needed to work on a large-scale project and that situated coordination is paramount, software developers and software teams need to rely on knowledge resources embedded within, available through, and derived from a network of relationships, also known as social capital. Social capital is both the network itself and the assets that may be mobilized through that network [22]. Oliver et al. [23] refers to a knowledge network as a network that facilitates learning and connects experts and novices to support on-demand, continuous learning through group interaction.

As mentioned before, teams in large-scale projects often work in isolation, while communicating and coordinating with the key experts in their network [7]. Therefore, coordination is needed on different levels: inside the team, among the teams, and between a team and the rest of the organization. In their systematic literature review, Mathieu et al. [24] conclude that a team's centrality in an inter-network is conducive to performance. Centrality appears to provide teams with advantages in terms of acquiring and applying resources. The team has social capital both in terms of how it can leverage the social interaction within the team, and how it uses the external contacts in its network to create value [25].

Social capital enables software teams to achieve results that would be impossible without it or could only be achieved at an extra cost for the development team [25]. Furthermore, because social capital increases the efficiency of information diffusion, a large-scale project can have less redundancy in e.g., skills or roles, if knowledge networks are cultivated; i.e., if the social capital is strong.

The value of social capital for a team and team members depends on a number of factors, including the knowledge network characteristics. For example, Burt [6] suggests that poor communication and coordination within an established network leads to poor performance. Groups working in isolation benefit least from external networking. In contrast, cohesive groups are able to circulate the information gained from the knowledge network and maximize their own performance. Characteristics of the knowledge network surrounding a team also influence the value of social capital. One such factor is external contact redundancy. The same contacts lead to the same knowledge sources, resulting in redundancy. Since networking requires time and effort, contact redundancy hinders performance. Thus, for two knowledge networks of equal size, the one with fewer redundant contacts will provide more benefits.

Network stability plays another important role. When people leave the organization, their connection usually dissolves with whatever social capital it contained [26]. It is fair to assume that other staff changes, e.g., team member rotations, promotions, and relocations, also affect the social capital of the team(s). Furthermore, different types of teams have different networking needs. Lewis [27] found that rich external knowledge networks are more valuable to cross-functional teams working on unfamiliar tasks, than to team members working on familiar and unrelated tasks. The main reason is that it may be less critical for team members in specialized teams, e.g., component teams, to integrate their expertise to perform well.

Evidently, a large-scale software development project enables the accumulation of social capital when its members are brought together to undertake their primary task, supervise activities, and coordinate work, particularly in contexts requiring mutual adjustment, i.e., coordination by informal communication [28]. The role of so-called "boundary spanners" is particularly important for connecting the remote parts of organizational networks [16,29,30]. Furthermore, training mechanisms exist to accumulate the social capital of software development teams at work. Certain development approaches (e.g., agile software development) and development practices (e.g., pair programming, daily meetings, and review meetings) foster frequent networking and extensive interaction inside the development teams. For example, Paasivaara and Lassenius [17] found that communities of practice (CoPs) and participation in different forums foster networking across development teams and units, as in large-scale development. Wohlin et al. [25] also emphasized that by investing in social capital, organizations can make the specialized or unique knowledge possessed by the few available to the many.

3. Research methodology

Our research is exploratory in nature [31]. To understand what influences team knowledge networks and networking behavior, we conducted a multi-case study of two large-scale software projects in two software companies, Ericsson and ABB, with software development teams as embedded units of analysis.

3.1. Case and subject selection

We conducted an embedded case study [31]. The unit of analysis was a development team in the context of a large-scale software project in a software-intensive company. As exploratory case-based research, our sample strategy was not to obtain accurate statistical evidence on the distribution of variables within a population, as in hypothesis-testing studies, but rather to rely on theoretical sampling [32]. From large-scale multi-site companies, we selected software projects and development teams with overlapping and complementary characteristics for the case study, to find a broader range of factors contributing to networking behavior.

Companies: The companies were selected based on convenience sampling. Both Ericsson and ABB have large-scale software development projects with company sites in Sweden and at least one offshore location. In addition, the two companies are of polar types when it comes to working methods (agile versus traditional). Polarity in case selection is likely to help extend the emergent findings [32].

Large-scale product development efforts: In each company, we selected one large-scale distributed software project that fulfilled the selection requirements, i.e., being a highly complex multi-team and multi-site endeavor. We evaluated complexity in terms of the knowledge demands, which were impossible to fulfill by

Table 1
Survey participants.

	Teams	Location	Total team members	Participated in the survey	Response rate
Ericsson	E-SWE-1	Sweden	8	6	75%
	E-SWE-2	Sweden	7	5	71%
	E-SWE-3	Sweden	6	5	83%
	E-CHN-1	China	9	8	89%
	E-CHN-2	China	5	5	100%
	E-CHN-3	China	6	6	100%
ABB	A-SWE1-1	Sweden 1	5	5	100%
	A-SWE1-2	Sweden 1	6	4	67%
	A-SWE2	Sweden 2	5	5	100%
	A-IND	India	18	12	67%
			75	**61**	**81%**

a single individual. Company representatives selected the candidate projects that fulfilled our requirements: multi-team, multi-site projects concerned with maintaining and evolving complex software systems.

Teams: In each company, for each large-scale software project, a selection of teams was identified and then analyzed. Since research activities required significant involvement from the teams and the researchers, we selected teams for analysis following the maximum-variation strategy [31]. In both companies, we selected teams from each of the main development locations. We selected mature as well as relatively new teams, and teams working with familiar and unfamiliar tasks. This sampling was done with the help of company representatives.

3.2. Data collection

As part of the case study, we gathered rich empirical data (see Table 2) from 65 survey responses, 31 interviews, 11 focus groups, and observations from visiting five different sites (two in Ericsson and three in ABB). To improve the validity of our findings, we strived for triangulation [31]. We collected data through different means (aiming for methodological triangulation) and from different roles in the development organizations (aiming for data source triangulation). We also performed member-checking [33], i.e., we organized feedback sessions with all the teams and the management, where we presented our findings to obtain feedback on our interpretation of the findings, and derived conclusions.

3.2.1. Survey

To answer our research question, we conducted a social network analysis survey to measure and capture the structural relations of team members inside and outside the team. The survey design was a partial replication of an empirical survey by Manteli et al. [30]. The social network survey is available in Appendix 1. We asked respondents to identify people that they sent project-related knowledge to, or retrieved knowledge from, as well as the nature and content of the knowledge transferred or retrieved. The respondents were asked to freely recall their contacts and fill in the details about each contact in writing (on paper or electronically). Based on this, we obtained a directed knowledge network, i.e., indicating the direction of the knowledge transfer, to or from the team.

Our sample included the teams that participated in the focus groups (see Table 2). The participation and response rates for the survey are given in Table 1. The networks included not only participants of the survey, but also non-participants recalled by the survey respondents, e.g., team members who did not participate, members of other teams, or those in supporting roles.

3.2.2. Interviews

Interviews were conducted in each company to understand the cases and the history. We interviewed representatives from all sites and included different roles (31 interviews in total, see Table 2 for details). The interviews were one hour long, on average, and all but one interview was conducted in person. All interviews were conducted in English and tape recorded. Interviews in Ericsson were transcribed, while interviews in ABB were analyzed using the recordings and notes written by the researchers.

3.2.3. Documentation

The qualitative data was supplemented by documentation (see Table 2) that helped us to understand the software development context, processes, and goals at both companies.

3.2.4. Observations

Onsite visits were organized to all onshore and offshore sites to better understand the environments in the different sites of the two companies. We visited Ericsson's main Swedish site and ABB's main Swedish site on multiple occasions. Two researchers paid a one-day visit to ABB's other Swedish site. One researcher spent one week in ABB's offshore site in India, and two researchers visited Ericsson's Chinese site for one week. The observations made during the visits were captured in the form of written notes.

3.2.5. Focus groups

After the interviews, focus groups with development teams and groups were organized in each of the companies. Focus groups are used to quickly obtain information on emerging phenomena through structured, moderated discussions with groups of practitioners [34]. More importantly, focus groups help tap into a group's collective memory by allowing participants to build on the responses of others, which leads to ideas and observations that might not emerge during individual interviews [34].

All focus groups followed a predefined structured agenda. In each focus group, we acquired information about the skills needed for solving the 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 the team's perception of their performance. The focus groups were moderated by the researchers, and were two to three hours long. All focus groups in both companies were recorded and later transcribed.

3.2.6. Feedback

We presented the findings to the companies and teams as a part of member-checking. Member-checking involved obtaining feedback on our understanding of the networks and the knowledge mobilized through these networks, our interpretations made during the data analysis, and perceived practical implications of the findings. Member-checking helps reduce many types of threats to validity, since both the respondents and the researcher can look through the material [35]. Presentation of the findings made the subjects feel as being part of the process. Presenting the findings is also very important in situations where the findings may have an impact on the way the practitioners work [36].

Interestingly, during the presentations, we received more feedback from the managers than the teams. The feedback contained explanations for certain findings, reflections on whether the findings confirmed the beliefs of the managers and team members, and what they thought was surprising or unexpected in the findings. We referred to the feedback received from the companies in

Table 2
Empirical data collection and analysis.

Ericsson					
Activity	Location	Time	Participants	Researchers*	Data gathered
Survey	Sweden	Apr 2014	3 Swedish feature teams, 16 participants	A1, A2, A3	Individual networks, purpose of communication, and availability scores
	China	Mar 2014	3 Chinese feature teams, 19 participants	A1, A2, A3	
Interviews	Sweden	Aug 2013	7 interviews, incl. an agile coach, planning manager, product manager, release manager, two design owners, system owner, system architect, and three technical experts (TARs).	A1, A2	Description of the system, roadmap planning, assignment of the tasks to the teams, mechanisms for quality control, and administrative coordination.
	China	Mar 2014	8 interviews, incl. an agile coach, team responsible manager, two operational product owners, TAR, previous system owner, and node architect. One Chinese TAR was interviewed by phone.	A1, A2	
Documentation	Sweden, China	May 2013–May 2014	System descriptions, specific role descriptions, list of team members, years of experience in the company, and process descriptions	–	Context of software development activities and team information
Observations	Sweden	Aug 2013–Sep 2014	Several office visits	A1, A2, A3, A4	Context information for the teams' work coordination, office layout
	China	Mar 2014	A week long office visit, observations of 5 daily Scrum meetings	A1, A2	
Focus groups	Sweden	Oct 2013	3 Swedish feature teams	A1, A2	Knowledge needs for feature development, presence of the knowledge in teams, external interaction, and externalized knowledge. Scores for reliance on different sources.
	China	Mar 2014	3 Chinese feature teams	A1, A2	

ABB					
Activity	Location	Time	Participants	Researchers*	Data gathered
Survey	Sweden 1	Feb 2014	2 teams, 9 participants	A1, A2, A3	Individual networks, purpose of communication, and availability scores
	Sweden 2	Feb 2014	1 team, 5 participants	A1, A2, A3	
	India	Jul 2014	Group of developers, 12 participants	A1, A2, A3	
Interviews	Sweden 1	Aug 2013	6 interviews, incl. the system architect, testing and support manager, product manager, team lead, project manager, and visiting senior developer from India.	A1, A2, A4	Description of the system, projects' planning, development processes, and mechanisms for quality control
	Sweden 2	Feb 2014	3 interviews, incl. a unit manager, project manager, and team lead	A1, A3	
	India	Jul 2014	8 interviews, incl. a unit manager (2 times), project manager, 2 technical coordinators, and 3 developers (junior, experienced, and regular).	A3	
Documentation	Sweden, India	Oct 2013–Sep 2014	System descriptions, specific role descriptions, list of team members, years of experience in the company, and process descriptions	–	Context of software development activities and team information
Observations	Sweden 1	Oct 2013	Office visit with an excursion, multiple visits	A1, A2, A3, A4	Context information for the teams' work coordination, office layout
	Sweden 2	Feb 2014	Office visit with an excursion	A1, A3	
	India	Jul 2014	Week-long office visit	A3	
Focus groups	Sweden 1	Feb 2014	2 teams in the main Swedish location	A1, A3	Knowledge needs for feature development, presence of the knowledge in teams, external interaction, and externalized knowledge. Scores for reliance on different sources.
	Sweden 2	Feb 2014	The only team in the other Swedish location	A1, A3	
	India	Jul 2014	Joint session with both groups of developers	A3	

* This column refers to the authors' participation in the data collection; the first author is A1, the second author is A2, and so forth.

the result narratives, whenever we found it useful to explain certain observations. Finally, we sent the manuscript of this paper to the company representatives to verify the accuracy of our description of the context.

3.3. Data analysis

We started with a within-case analysis, which was conducted on multiple levels—team, site, and organizational—before moving to a cross-company analysis, as suggested by Eisenhardt [32]. First, we constructed knowledge networks for each team based on the survey data. Special preparations were performed to analyze the survey data.

Survey data preparation was needed to ensure the reliability and quality of the obtained dataset. The first and third authors carefully screened the individual responses, which included clarifying the names and roles of each network contact, clarifying unclear responses (e.g., unknown abbreviations of roles, processes, or subsystems), and removing invalid responses. We also merged recip-

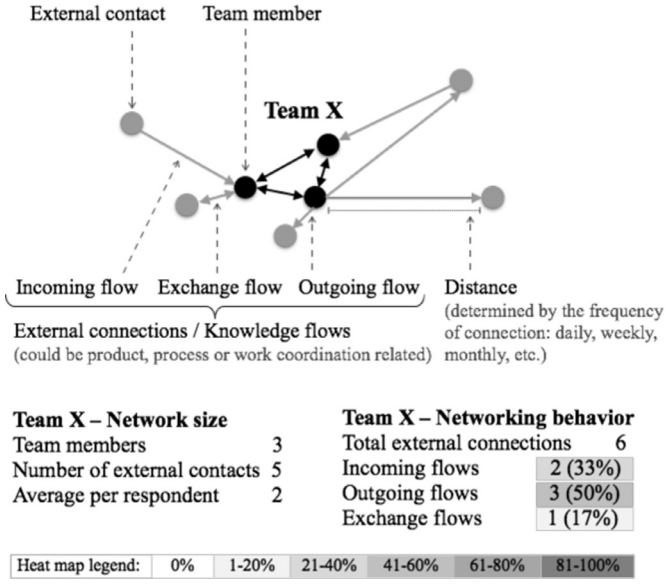

Fig. 1. Example: Team network visualization and measures.

rocal relations. In a reciprocal relation, Person A identifies Person B as a knowledge-sharing contact, and Person B likewise identifies Person A. Both Person A and Person B reported the purpose of the knowledge relation between each other; we merged them by semantically combining the answers and averaging the interaction frequency. Finally, we transformed the data 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 the survey responses, i.e., who referred to whom.

Survey data analysis aimed to understand an individual team's networking behavior. We started by deriving the purpose of the knowledge flows. During this process, we learned that respondents regarded very different information types as the knowledge important for their job. Hence, we classified the responses into the following categories:

- Product-related knowledge includes knowledge about program properties, existing architecture, concept location within the code, etc. [29];
- Process-related knowledge includes knowledge about coding conventions, development tools, ways of working, and skills [29]; and
- Project-related work coordination includes information about project milestones, delivery schedules, progress updates, reviews, and inspections [37].

In Fig. 1, an example of the outcome of our analysis is visualized. For each team, we calculated the total number of external contacts and external connections (knowledge flows). We noted the purpose of the knowledge flows as being related to the product knowledge, process knowledge, or work coordination. In our analysis, we also separated incoming, outgoing, and exchange flows. Finally, we constructed table summaries, in which we visualized the most and least common flows as a heat map (the frequency increase of a knowledge flow is visualized by a darker color of table cell (see our example in Fig. 1).

Survey data visualization was performed using Gephi,[1] which is an open source software for visualization and analysis of social networks. Network graph layouts were constructed using the Force-Atlas-2 algorithm, which is a simple spatialization algorithm for large network visualizaiton proposed by the Gephi team.

Within-case analysis included a qualitative analysis of the transcribed data and aimed to better understand teams' knowledge networks and networking behavior. Based on the focus group data, teams' self-reported characteristics, observation notes, and survey data analysis (as described above), we created detailed case write-ups, which were central to the generation of insights [32]. The companies' representatives verified the accuracy of the narratives.

Cross-case analysis targeted cross-team, cross-site, and then cross-company comparisons. We started by comparing the constructed knowledge networks, and then thematically analyzed team descriptions in search of patterns. Since the way team worked varied across sites, and even more across companies, we also created project-level write-ups. The cross-company analysis was highly iterative and led to refinements of the concept definitions due to differences in terminology across teams, sites, and companies, and because related categories could be merged under more general concepts. This is not uncommon in pattern generation according to Eisenhardt [32] who suggests that researchers shall apply cross-case searching tactics to look at the different categories or dimensions, determine similarities and differences, and detect the emerging patterns.

For example, what later emerged as "having an expert in the team" was first identified in Ericsson's case as technical area responsible developers (TARs), who acted as knowledge hubs and were in some cases members of a team. TARs had a set of formal responsibilities and dedicated time for networking. ABB had no formal experts in the same sense as in Ericsson; thus, our analysis of the expert structure was triggered by the findings from the cross-company analysis. We performed an additional inquiry and learned that ABB teams rely primarily on informal experts who do not have formal role descriptions; only after we presented our preliminary findings was the formal role of Technical Coordinators introduced in the Indian site.

The emergent explanations and determinants of the team networks were summarized in the form of team profiles (see Tables 3 and 6 in the findings). When profiling the teams, we also compared the emergent concepts with existing research that identified what determines the external network size and networking behavior of a team. Some factors were consonant with related literature, e.g., company experience [37], task complexity, and task familiarity [6], while other factors were case-specific.

To help with hypothesis generation [32], we summarized our findings in the form of a relation model, in which emergent factors are linked with the teams' external knowledge networks and networking behavior (Fig. 5). The factors included in the model came from multiple data sources (mentioned by more than one team).

4. Findings

In this section, we present our findings in the two case projects. In each case, we start with a detailed project description (the Ericsson case in Section 4.1 and the ABB case in Section 4.3). For each company, we describe the product under development, the software development process, and how the teams were organized. Case descriptions serve the purpose of putting our findings in context; thus, clarifying the validity and applicability of our conclusions. We continue by presenting the teams' knowledge networks, network characteristics, and knowledge flows in light of the context (the Ericsson case in Section 4.2 and the ABB case in Section 4.4). We conclude our findings with a cross-company analysis (Section 4.5).

[1] Gephi is maintained by Gephi Consortium, https://gephi.github.io.

Table 3
Teams' profiles.

Teams	Location	Team members	Expert in the team	Company experience (average)	Team experience (average)	Type of team	Task familiarity	Task complexity	Approach to solving simple or familiar tasks	Approach to solving complex or unfamiliar tasks	Participation in forums and CoPs
E-SWE-1	Sweden	8	No	12.3 years	2.1 years	Cross-functional feature team	Unfamiliar	High	Human capital	Social capital	Often
E-SWE-2	Sweden	7	Yes	9.6 years	1.6 years		Varying	Varying	Social capital	Social capital	Sometimes
E-SWE-3	Sweden	6	Yes	13.3 years	2.3 years		Familiar	Varying	Human capital	Org. capital	Sometimes
E-CHN-1	China	9	Yes	2 years	2 years		Familiar	Low	Human capital	Social capital	Seldom
E-CHN-2	China	5	No	2.8 years	2.2 years		Familiar	Low	Human capital	Social capital	Seldom
E-CHN-3	China	6	No	2.2 years	0.9 years		Unfamiliar	Low	Human capital	Social capital	Seldom

4.1. Ericsson case description

Ericsson is a leading provider of telecommunication systems and equipment for mobile and fixed network operators. The company develops generic software products offered to an open market and complex compound systems with customized versions.

4.1.1. Studied project

The project under study was a distributed development effort of one of 30 sub-systems belonging to a large software-intensive product accounting for more than many-million lines of code written in different programming languages. The studied product was on the market for more than five years, and at the time of investigation contained 14 components and interacted with eight different sub-systems. The number of developers working on the project grew from eight developers in 2007 to 30 developers in 2009, and scaled up to around 60 developers by 2013. In early 2014, there were five in-house teams in Sweden, eight in-house teams and two outsourcing teams in China, as well as two in-house Korean teams.

4.1.2. Development approach and knowledge needs

Agile and Scrum have been practiced since 2008, and the project was organized in seventeen self-managing cross-functional feature teams comprised of members with different roles. In most cases, features were handled by one team, and represented a piece of functionality or a requirement requested by the system owner. Features can address enhancements required by the telecommunication standards, customer specific requirements, and evolution-based enhancements. Notably, features are domain specific; thus, particular domain knowledge is needed to be able to perform a task. Over time, a team develops a specialization in a certain type of features, which can be associated with, but are not limited to, system functionality, system layers, or components. Features also differ by complexity. The most complex features require changes in several parts of the complete system. Implementing such complex features puts high demands on the team and the knowledge possessed by or accessible to them. Occasionally, an existing team changes its specialization and learns a new type of features.

Bug fixing is a rotating task. A team may spend up to half a year on bug-fixing, simultaneously broadening their expertise, because they must work in several parts of the complete system when fixing a bug. In addition, teams working for 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. A team's work on a feature starts by receiving a high-level description and continues with a so-called "sprint zero," in which team members work closely together on designing the feature, followed by the demo of the feature design, development, and delivery. Technical-area-responsible developers (TARs) play an important role in consulting feature teams and approving feature solutions. These are the most senior developers responsible for a system area, and spend 50% of their time sharing their knowledge, supporting teams on technical questions, performing code reviews, and approving feature solutions. The rest of the time, a TAR is a team member.

4.1.3. Teams and teamwork characteristics

We studied six Ericsson teams—three from the Swedish site (E-SWE-1, E-SWE-2, and E-SWE-3) and three from the company's Chinese site (E-CHN-1, E-CHN-2, and E-CHN-3). The team profiles are presented in Table 3.

Experience: From the profiles, we observe that the Swedish teams can be generally characterized by their long company experience (around 10 years on average). The experience of working together in the team varies. In contrast to the Swedish teams, the Chinese teams are composed of members with less company experience (only two to three years on average). Two of the Chinese teams have had occasional member changes and have operated together for two years, while team E-CHN-3 is the most immature team, with less than a year's joint work experience. Finally, two Swedish teams (E-SWE-2 and E-SWE-3) and one Chinese team (E-CHN-1) have TARs working 50% in the team.

Approach to solving tasks: Most teams are working with familiar tasks, and either rely on their own knowledge and skills (human capital) or teamwork (social capital). Teamwork and networking with experts are paramount to overcoming complexity or a lack of knowledge. Team E-SWE-1 is an experienced team that has been recently assigned a new type of feature and is, therefore, currently performing unfamiliar tasks. Team E-SWE-2 is working toward a specific customer, meaning that they need to work on a large part of the system; hence, the task familiarity varies. Team E-CHN-3 is a new team; therefore, the tasks are unfamiliar for most of the team members.

Task complexity also differs. The Chinese teams and Team E-SWE-3 are working on relatively simple features, while Teams E-SWE-1, E-SWE-2, and part of E-SWE-3 are assigned more complex tasks. While teams E-SWE-1, E-SWE-2, E-CHN-1, E-CHN-2, and E-CHN-3 regarded team discussions and networking with others as their main approach to problem solving, Team E-SWE-3 revealed that communication played a background role. In areas where team members were lacking knowledge, they primarily relied on documentation and source code (organizational capital), while relying on individual skills when working on familiar or simple tasks.

Participation in forums and CoPs: Finally, we asked the teams to describe their participation in different forums and communities of practice (CoPs) to learn about their networking habits. We found that the Swedish teams have a tradition of attending disciplined CoPs, although some teams are more active than others, while in China such forums are less attended.

The network graphs from the social-network analysis survey at Ericsson are shown in Fig. 2. The black dots and lines in the figure represent respective team members and team internal connections, while the grey dots are team contacts—supporting roles and other team members from any location. External network size and knowledge flows can be found in Tables 3 and 4.

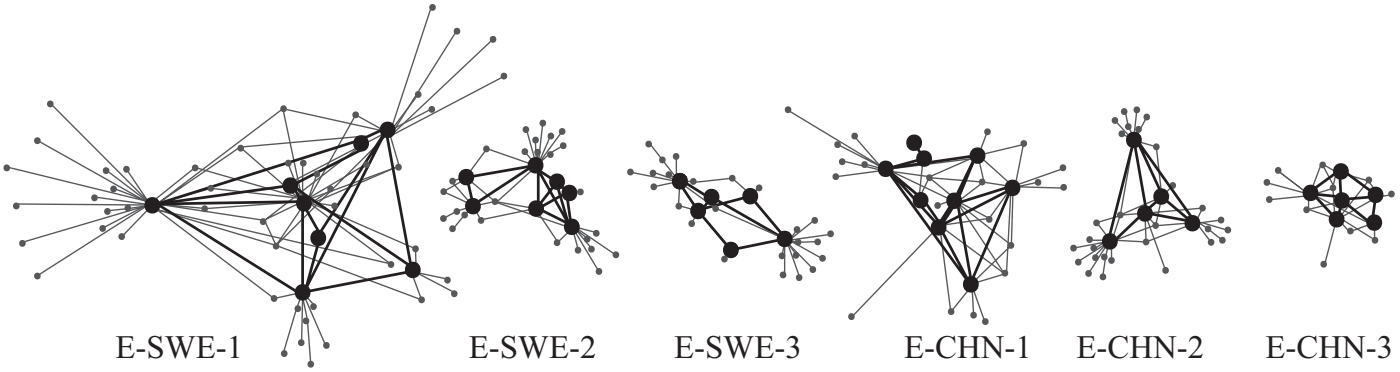

Fig. 2. Ericsson teams' knowledge networks.

Table 4
Teams' network characteristics.

Team	Number of team members	Respondents	External network size	
			Total	Average per respondent
E-SWE-1	8	6	48	13.2
E-SWE-2	7	5	26	5.8
E-SWE-3	6	5	25	5.6
E-CHN-1	9	8	21	4.9
E-CHN-2	5	5	29	7.6
E-CHN-3	6	6	11	3.3

Table 5
Teams' knowledge flows – networking patterns.

Teams	Total flows	Product knowledge flows			Process knowledge flows			Work coordination flows		
		Incoming	Outgoing	Exchange	Incoming	Outgoing	Exchange	Incoming	Outgoing	Exchange
E-SWE-1	67	22%	-	1%	7%	1%	37%	-	-	30%
E-SWE-2	36	14%	22%	11%	14%	-	22%	2%	-	14%
E-SWE-3	30	3%	27%	13%	1%	-	17%	7%	-	23%
E-CHN-1	52	30%	2%	4%	6%	-	8%	4%	-	46%
E-CHN-2	41	36%	-	2%	19%	2%	15%	2%	-	22%
E-CHN-3	26	65%	-	-	23%	-	4%	4%	-	4%

Heat map legend: 0% | 1–20% | 21–40% | 41–60% | 61–80% | 81–100%

4.2. Networking behavior in the Ericsson case

In this sub-section, we present our findings regarding the team networking behavior. We describe the team external network measures: network size and networking patterns.

4.2.1. External network size

From the external network size measure, we learn that Swedish Team E-SWE-1 has the largest network, Chinese Team E-CHN-3 has the smallest network, while the other teams have approximately the same average network size. Teams have different contacts in their networks. Evidently, software teams in large-scale projects at Ericsson are supported by a large number of roles, including architects, TARs, informal technical experts, line managers, product owners, operative product owners, system managers, configuration managers, testing-framework experts, continuous-integration experts, agile coaches, and integration leaders. Further, teams also interact with members of other teams. However, when asked about the expected team external-network size during the feedback sessions, the managers at Ericsson suggested it would be around 10 contacts. In fact, since the amount of networking was underestimated, the company environment did not satisfy all the needs of the teams. In the focus groups, teams complained about the lack of dedicated rooms and inability to have ad hoc meetings due to working in an open-space environment and an unwillingness to disturb others at work. As a developer from Team E-SWE-1 said: *"If you can't book a meeting [well in advance], it becomes harder to [collaborate]."*

The selection of teams in different locations also allowed us to compare teams with different backgrounds. In focus groups with the teams and feedback sessions with the Chinese managers, we received a few explanations of networking in China pointing to cultural differences. Chinese teams were said to prioritize networking within the team over interactions with people outside the team, because supporting their own team was seen as paramount. The developers explained that delivering value for the company in the form of good quality and new features was seen as most important, and networking not directly supporting this goal was not a priority. This is also evidenced in the required improvements, as reported by Team E-CHN-2 and E-CHN-3. Both teams confessed that they should work more on interacting with people outside of the team, while E-CHN-1 stated that they do not interact much with other teams beyond the joint-feature work. Language barriers between China and Sweden, some people being shy, and newly hired people without a network were given as additional explanations for why some Chinese team members had a small network. This suggests that networking practices could be rooted in culture.

4.2.2. Networking patterns

When analyzing knowledge flows (how the network is used), we found that teams depended on diverse knowledge pulled, pushed, and shared in their networks, i.e. product knowledge, process knowledge, and knowledge related to work coordination. In Table 5, we classify the teams' knowledge flows in terms of the networking purpose and use a heat map to distinguish between the patterns— darker colors show the most common flows and lighter colors show the least common flows. Note that we use percentages, and thus do not compare the actual number of flows across teams. Moreover, we cannot say anything about the amount of communication, since some of the contacts in the network are more frequent than others.

Scouting for product knowledge in new teams: Team E-CHN-3 was a newly founded team. During the team retrospective, many referred to the team as consisting of young graduates with little or no experience, meaning there was not much knowledge or connections in the organization. As someone said: *"We know we*

should communicate with other people, but we don't know how to communicate efficiently." As a consequence, their network was small, there were no outgoing knowledge flows and the vast majority of the reported knowledge flows were incoming (65%). The team members reported receiving product-related knowledge, tips, and tricks from their peers from other teams, as well as procedural directives and suggestions from supporting roles.

Scouting for product knowledge in mature teams: Due to the large product size and changing needs for development efforts, teams need to learn new product areas. As someone said, *"There are around 30 sub-systems [in the complete product] and two-three engineers have worked in 20 subsystems. They are exceptions. Most are working in four-five… I would say that no one in the organization, even senior architects, understands the complete product."* We found that even the team with long company experience (Team E-SWE-1) depended on product knowledge pulled from the network of formal and informal experts when working on complex or unfamiliar tasks. Teams with short company experience (Teams E-CHN-1, E-CHN-2, and particularly E-CHN-3) scouted for product knowledge to perform the tasks, usually approaching the formal experts (TARs). Developers with limited experience mentioned seeking knowledge about the sub-system's architecture and code, knowledge about related sub-systems, and knowledge of different telecommunication protocols and standards. Evidently, all teams that reported teamwork and networking as their primary approach to solving complex tasks reported a relatively high number of incoming-product knowledge flows (22% for E-SWE-1, 30% for E-CHN-1, 36% for E-CHN-2 and 65% for E-CHN-3), while Team E-SWE-3, which relied on documentation and source code, did not (only 3% of all networking connections were incoming knowledge flows).

Bridging product knowledge: Formal technical experts (TARs) in the team play the role of bridges and product-knowledge hubs for other teams, and are the ones behind the outgoing flows in Teams E-SWE-2, E-SWE-3, and E-CHN-1. They provide product knowledge for the teams working on modules, which are part of their responsibility. From the survey, we found that teams with a TAR scouted less for product knowledge, indicating that when a TAR was part of the team, he or she acted as a knowledge hub for the team.

TARs were found to be critical for the Chinese teams. A member from Team E-CHN-2 commented: *"For us designers, the most frequent persons we interact with outside the team are the TARs."* Simultaneously, we found that TARs could delay teams when the experts disagreed. As a member from Team E-SWE-1 stated: *"When you have different TARs, they have different opinions."* Team E-SWE-1 discussed a number of frictions with the TARs, which demonstrated that the TARs have questioned their competence and delayed their progress. When asked what would happen if TARs did not exist, someone from Team E-SWE-1 said, *"The code would be less structured,"* and another commented, *"I think that we would survive that change; I think that we would be better,"* and another said, *"Will be faster."*

Receiving process-related advice: Team E-CHN-3 reported receiving a lot of guidance from the line managers, product owners and other roles in terms of how to perform their work (23% of all networking connections). Despite the availability of documentation, those who are new to the company corporate culture and ways of working are often mentored and receive a lot of advice. As a member of Team E-CHN-3 said: *"I am adjusting to this new organization. The first week I' m here, I read quite a lot of documents about the process […] So, probably, at this point, I got a lot of knowledge. But not all the knowledge"*, and another member continues *"Yes, several teams have told us the same. It's a lot of communication with different [roles]"* .

Sharing know-how: The Swedish teams reported having a large amount of networking dedicated to exchanging know-how information and discussing ways to work with people in other teams (37% in E-SWE-1, 22% in E-SWE-2 and 17% in E-SWE-3). This mostly happens in disciplined forums as CoPs, and through personal links. One reason for the need to discuss working methods is that the process information is spread across multiple systems. This can be illustrated by the following quote: *"… you often get [updates] in the mail and if you are happy or lucky then they are also updated on a webpage or a wiki page somewhere. Itś very hard to [find the information]."*

Work-related knowledge coordination: Work coordination occupied a substantial part of networking involving product owners, other teams, and in the case of the Chinese teams, system analysts who work outside the teams (above 20% in four out of six teams, and as high as 46% in E-CHN-1). The coordination was related to requirements, technical and managerial dependencies, and deliverables. For example, several members from Team E-SWE-1 maintained links with the supporting teams and roles responsible for testing processes and tools by reporting problems, obtaining updates, and providing feedback on solutions. Team E-SWE-2 and E-CHN-3 had less such networking (14% and 4% respectively). Team E-SWE-2 was working towards a specific customer and less with others, while Team E-CHN-3 was new and thus worked in relative isolation.

4.3. ABB case description

ABB is a large international company developing complex embedded systems for power and automation solutions. Systems in the safety-critical portfolio are developed following a traditional plan-driven software development methodology, in which predictability and control are important.

4.3.1. Studied project

The project under study was a complex software-intensive system containing a number of components consisting of many-million lines of code written in several programming languages. At the time of investigation, the system had been evolving for over a decade. In early 2014, the work was distributed across two sites in Sweden and one site in India, all branches of ABB. The Swedish site, referred to as Sweden 1 in the findings, is responsible for the product. In general, there are six software teams at the main development site in Sweden (referred to as Sweden 1), a software team in the second site in Sweden (referred to as Sweden 2), 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.

4.3.2. Development approach and knowledge needs

At ABB, heavy emphasis is placed on product and process quality, and a V-model development methodology is put in practice, in which tasks are structured and project members' roles are task-specific. Development work is organized in projects, which can include new system generation, new functionality development, roll-ups of large maintenance projects, and pure maintenance projects. The teams are disciplined teams comprised of programmers and led by a team lead. Each team specializes in a particular part of the system, i.e., a system component. A good developer is said to have only a 50% overview of the system. One system architect said: "*It requires at least three years to get to know the code base."* Team members may be involved simultaneously in several projects. The team responsible for the corresponding part of the system usually fixes the bugs. 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 to locate the bug and fix it. This is mainly due to the system knowledge differentiated across multiple disciplined teams.

Table 6
Teams' profiles.

Teams	Location	Team / group members	Expert in the team	Company experience (average)	Type of team	Task familiarity	Task complexity	Approach to solving simple or familiar tasks	Approach to solving complex or unfamiliar tasks	Participation in forums and CoPs
A-SWE1-1	Sweden 1	5	No	8 years	Disciplined component teams / group	Familiar	Medium	Org. capital	Org. capital	No
A-SWE1-2	Sweden 1	6	Yes	11.3 years		Familiar	Medium	Social capital	Social capital	No
A-SWE2	Sweden 2	5	Yes	17.2 years		Familiar	Medium	Org. capital	Org. capital	Seldom
A-IND	India	18	Yes	4.6 years		Unfamiliar	Low	Human capital	Social capital	No

ABB practices detailed up-front design in which only selected team members are involved, forming a virtual design team. When the development projects start, the work is usually well specified and coordinated through team leads, who further assign the tasks to individual team members. The role of each site is determined by the component specialization. Software projects are usually led from Sweden 1 (the main location), while hardware projects are led from Sweden 2. There are a number of senior developers in both Swedish locations who are frequently contacted for consultation. To support cross-site coordination, the most senior developers in India have been appointed as technical coordinators.

4.3.3. Teams and teamwork characteristics

In ABB, we studied three Swedish teams from two locations (Team A-SWE1-1, A-SWE1-2, and A-SWE2) and a large group of developers in India (A-IND). It is important to note that the Indian group is not formed as a team, but rather as a working group. Therefore, the team-level findings from A-IND are not directly comparable with those from the teams. The units studied in ABB are profiled in Table 6.

Experience: Developer experience at ABB differs across locations: the company experience in Sweden is significantly higher than that in India (approx. 8, 11, and 17 years on average versus less than 5 years).

Tasks: Work organization within the teams differs. In teams A-SWE1-1 and A-SWE2, team members know a certain part of the code and usually only receive tasks within their own specialization. Redundant knowledge in these teams is therefore rare, and collaborative development work is generally not common. In contrast, members of Team A-SWE1-2 are expected to be able to work on diverse tasks, and knowledge redundancy is therefore built intentionally. This subsequently requires more collaboration among the team members. The complexity of the tasks assigned to Swedish teams is perceived to be of medium difficulty and most of these tasks are familiar for the teams.

Indian developers are responsible for the development and maintenance of two types of software protocols that are responsible for interaction between components; this work is regarded as low complexity. We also learned that software protocol work is quite isolated and requires only one or two developers per protocol. Each protocol might be completely different, which means that the knowledge needed to develop a protocol is more specialized and cannot be fully reused or useful for others. However, we learned that developers often changed their specialization, and knowledge redundancy was built intentionally, to prevent the loss of knowledge due to employee turnover. Given the frequent staff changes, the level of familiarity with the tasks was said to be low.

Reliance: During the focus groups, we learned that when solving complex tasks, the teams relied on a variety of strategies. Teams A-SWE1-1 and A-SWE2 primarily relied on the source code and documentation, i.e., organizational capital, while Team A-SWE1-2 and Group A-IND relied on teamwork and networking, i.e., social capital. Interestingly, most Indian developers perceived that the importance of social capital grows as the task complexity increases, admitting that simple tasks could be handled relying on their own skills, i.e., human capital.

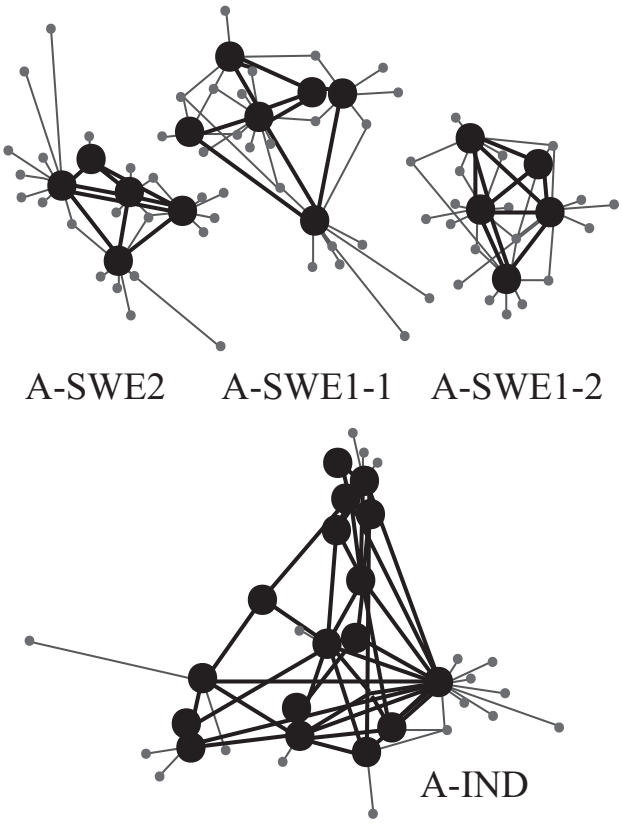

Fig. 3. ABB teams' knowledge networks.

Table 7
Teams' network characteristics.

Teams	Team members	Respondents	External network size	
			Total	Average per respondent
A-SWE1-1	5	5	19	6.4
A-SWE1-2	6	4	23	7.8
A-SWE2	5	5	28	5.6
A-IND	18	12	14	1.7

Participation in forums and CoPs: During the interviews, we learned that there were no institutionalized regular meeting forums for development teams. However, Team A-SWE2 mentioned several interest groups, e.g., an architecture group and static analysis group, in which those interested occasionally organized seminars to discuss, e.g., tools, to be used in development.

The team networks based on the survey data can be found in Fig. 3, and network characteristics are in Table 7.

Table 8
Teams' knowledge flows – networking patterns.

Teams	Total flows	Product knowledge flows			Process knowledge flows			Work coordination flows		
		Incoming	Outgoing	Exchange	Incoming	Outgoing	Exchange	Incoming	Outgoing	Exchange
A-SWE1-1	30	10%	-	33%	-	-	10%	-	3%	43%
A-SWE1-2	35	3%	-	54%	-	-	3%	-	-	40%
A-SWE2	28	29%	11%	21%	11%	4%	-	7%	4%	14%
A-IND	17	29%	-	29%	18%	-	12%	-	-	12%

Heat map legend: 0% | 1–20% | 21–40% | 41–60% | 61–80% | 81–100%

4.4. ABB networking behavior

4.4.1. External network size

We found that the network sizes at ABB were comparable to those reported in Ericsson (based on the average contacts per respondent), and so were the expected team external-network sizes, i.e., the managers suggested they would include around 10 contacts. In practice, team members in ABB interacted with project managers, product managers, safety engineers, configuration managers, technical experts, testers, and members of other teams. When looking at cross-site connections, it is worth noting that these were found primarily between the two Swedish sites. We also observed a trend towards the central development location. Teams A-SWE1-1 and A-SWE1-2, situated in the main Swedish site, primarily had local contacts, while the number of remote contacts in Team A-SWE2 and Group A-IND was higher than the number of local contacts. Notably, the Indian group in general had a small external network, and mostly networked internally.

4.4.2. Networking patterns

The classification of knowledge flows at ABB, in terms of product knowledge, process knowledge, and knowledge related to work coordination, is summarized in Table 8. The identified networking patterns at ABB are described next.

Sharing product knowledge: Networking at ABB primarily concerns product-related knowledge exchange (above 20% of all networking connections and as high as 54% for A-SWE1-2). In particular, we learned that members from different teams exchanged product-related knowledge regarding interdependent components. Team A-SWE1-2 reported that they communicated frequently with one other team and discussed component configuration, bugs, communication parameters, and intended product behavior. The team lead from Team A-SWE1-1 commented on the need for collaboration related to product interfaces: *"I think almost all tasks require interaction with others in the team or with other teams in [the same location]. That is a common method of working and it works nicely."*

Coordination also happens across sites. A member of Team A-SWE1-2 explained: *"We have the [A-SWE2] team in [Sweden 2] and also two other teams, so we have a lot of communication between [two components]. So sometimes, we have to discuss how the system should behave."* Cross-site interaction is supported by video-conferencing facilities; however, the teams confessed that it was difficult to have unplanned meetings since the video-conference rooms were frequently fully booked.

Scouting for product knowledge: was more important in the second development sites (29% of all connections in A-SWE2 and 29% in A-IND), and not in the main Swedish site (Sweden 1). This might be an impact of distance, when offshore sites perceive themselves more on the receiving end when requesting knowledge via electronic means of communication, while the onshore employees have a perception of mutually exchange the knowledge in a discussion or a face-to-face meeting. Interestingly, all of the Swedish teams reported receiving questions from India and sharing their product knowledge. In Team A-SWE2, this even required one dedicated engineer.

Work-related knowledge coordination: We found work-coordination exchange flows to be an important part of networking, primarily in Teams A-SWE1-1 (43%) and A-SWE1-2 (40%) from the main Swedish site. These teams reported a necessity to coordinate with other teams in the same location that were developing hardware and low-level software.

4.5. Cross-company analysis

In this section, we analyze the findings from the two cases in light of the context similarities and differences.

Both Ericsson and ABB had large-scale software development projects involving Sweden and at least one company-internal offshore location. While the sourcing setup was the same, we found differences in the ways of working and the team setup. The project studied in Ericsson followed agile methods, while the project studied in ABB followed a more traditional plan-driven software-development approach. Ericsson employed cross-functional feature teams, while ABB employed disciplined component teams (teams consisted of programmers, while testers and other roles were allocated outside of the team).

While the software development method and the task strategy varied, we found that the team knowledge networks varied more within the same large-scale software project than across the companies. The networks studied in Ericsson and ABB were generally quite similar in terms of network size (as seen in Tables 4 and 7). In our feedback sessions, managers from both companies estimated that a team would have around 10 contacts in their networks. Our findings surprised them. We found that most of the teams in both companies had six to eight contacts reported per respondent with half of these contacts being unique, resulting in a two and a half times larger external network than the managers were aware of.

While the number of years in the company (see team profiles in Tables 3 and 6) clearly influenced the social network of an individual, we found people with long experience having small networks and people with short company and team experience interacting with a large network. One Ericsson team (Team E-SWE-1), with long company experience and working on complex and unfamiliar tasks, had an exceptionally large network (13 contacts and 11 knowledge flows reported per respondent). None of the teams in ABB had a comparably large network. However, we found teams with small networks in both companies; e.g., the newly established Chinese team in Ericsson (Team E-CHN-3) and the relatively isolated group of Indian developers in ABB (Group A-IND).

While the managers were not aware of the role and importance of the teams' networks, both organizations had several mechanisms to enable and strengthen networking. During the focus groups, we obtained the team opinion and feedback on the efficiency of these mechanisms:

- ***Technical experts:*** Both formal and informal experts supported the teams when solving tasks, including the TARs in Ericsson who typically spent 50% of their time supporting other teams, Technical Coordinators in the Indian site at ABB, and senior developers in Sweden who supported other developers. These technical experts had a very large network and acted as bridge-heads.
  While teams within and across the two Swedish sites at ABB interacted actively, we learned that the absence of appointed

experts led to miscommunication and delays in coordination with the offshore site, because the Indian developers did not know whom to contact in Sweden. While there was little direct communication between the developers in India and Sweden, Technical Coordinators were introduced in India to act as liaisons, who maintained continuous communication with the remote sites and spread the awareness of who the Swedish experts were.

While there are many benefits of having appointed experts, we learned that there are also a few challenges. For example, an experienced team might not necessarily benefit from the necessity to seek formal expert approval on all their decisions. An experienced team with a large network and confidence in its own ability to provide a high quality solution, complained that TARs decreased their efficiency and autonomy. Another potential drawback of a formal expert structure is that a team with an expert member might rely on the expert's knowledge network, and, hence, the team members would interact less on their own. In our study, this was particularly evidenced in ABB, where one senior developer funneled the information between the Swedish teams and the Indian group (see the network in Fig. 3 and networking patterns in Table 8).

- ***Forums and CoPs:*** Formation of disciplined contacts by team members in Ericsson were facilitated through CoPs, in which team members came in contact with their peers working in the same area. In light of our research, we learned that it was also an important mechanism to establish non-redundant network contacts for individual team members. Ericsson established meeting forums for scrum masters, system analysts, programmers, and testers. However, membership and involvement in these CoPs were not enforced. Among the six studied teams, we learned that the Chinese teams participated less in CoPs, while the popularity of these forums varied in Sweden (see team profiles in Table 3).

  In ABB, we found that seminars and ad hoc interest groups were introduced on a need basis, but the frequency of these events was lower than those in Ericsson (see team profiles in Table 6).
- ***Communication infrastructure*:** Both companies established a communication infrastructure to support networking. However, we found that the infrastructure did not fully satisfy the networking needs, perhaps because the amount of networking was underestimated. We learned that Ericsson teams complained about the limited availability of meeting rooms in Sweden. In ABB, we found that informal networking within the same site was not seen as a problem, while networking across sites was impeded by the unavailability of video-conference facilities.
- ***Dedicated time:*** The Ericsson teams from Sweden complained about the limited time for networking, particularly in light of agile working methods, which dictate a fast development pace, leaving little time for other activities, such as CoPs. The high development speed was said to constrain team members who wanted to seek help from others, as well as those who were expected to spare some of their time for sharing their knowledge with others. It was also mentioned that priority was given primarily to mandatory interactions and meetings, while curiosity and knowledge exchange beyond the daily work could have been improved. Similarly, some teams confessed that they sometimes did not want to disturb others with questions. When we discussed this with the management representatives, they admitted that the teams' networking time was not being planned. However, this issue was partially addressed when it came to experts. In Ericsson, management dedicated 50% of the TARs' time to helping others, i.e., networking. Notably, time constraints were not mentioned as an issue in ABB.

5. Discussion

For decades, software development was seen as a profession with exceptionally low socialization needs; however, more contemporary research in this field argues otherwise [38]. We have studied software teams in two projects and their use of knowledge resources available through a network of relationships in two large companies, Ericsson and ABB, which developed software-intensive products that originated in Sweden and then became large-scale distributed endeavors.

We learned that when solving complex or unfamiliar tasks, most development teams in large-scale projects relied on social capital, i.e., teamwork and external team networking. Our findings contribute to the debate on how teams are configured and the need for providing support for social integration as a factor leading to superior performance [37]. Thus, while expertise is a necessary input, its presence in the team or in the team's network is not sufficient to affect performance if team members cannot coordinate their expertise within the team or with their external network. The importance of social capital and its ability to compensate for gaps in individual knowledge and skills also reflect ideas recently proposed as a General Theory of Software Engineering [25].

In the following, we discuss the cases in light of our research question:

What influences team knowledge networks and networking behavior in large-scale projects?

We start by discussing factors that influence teams' network size, summarize networking patterns observed in the cases, and then conclude with some practical implications emerging from our findings. Our findings are also illustrated in a relation model (see Fig. 4). It is important to note that the intention of the model is to enable generation of hypotheses based on our findings. The main hypothesis is that the size of a team's external knowledge network affects its networking behavior. Also the relationships between the impact factors (company experience, turnover of the employees, task profile, etc.) deserve research attention and further validation. In other words, further research and complementary cases are required to validate the relationships hypothesized in the current version of the model.

5.1. Factors that influence network size and networking behavior

Based on our findings, we determined that the following factors might influence the size of team networks:

- ***Company experience*** affects the size of an individual network. Team members need to be familiar enough with each other's experiences, skills, and specialized knowledge to facilitate the emergence of expertise coordination processes [37]. Previous research also indicates that because new teams with many newly hired people will likely not know whom to contact, their networks may be smaller Rus and Lindvall [39]. This might be an alternative explanation for why networking varies across countries, considering that some locations have demonstrated a high turnover factor.
- In our study, the Chinese and Indian sites were said to have high ***employee turnover*** in comparison with the Swedish sites, and thus did not accumulate the same number of contacts. Similarly, related research warns that when people leave an organization, their social connection usually dissolves with whatever social capital it contained [26].
- ***Participation in forums and CoPs*** potentially increases the amount and frequency of communication between teams, and therefore the network size. This is an important source of contacts and a motivator for process-related knowledge exchange, e.g., know-how and skills and learning who the informal tech-

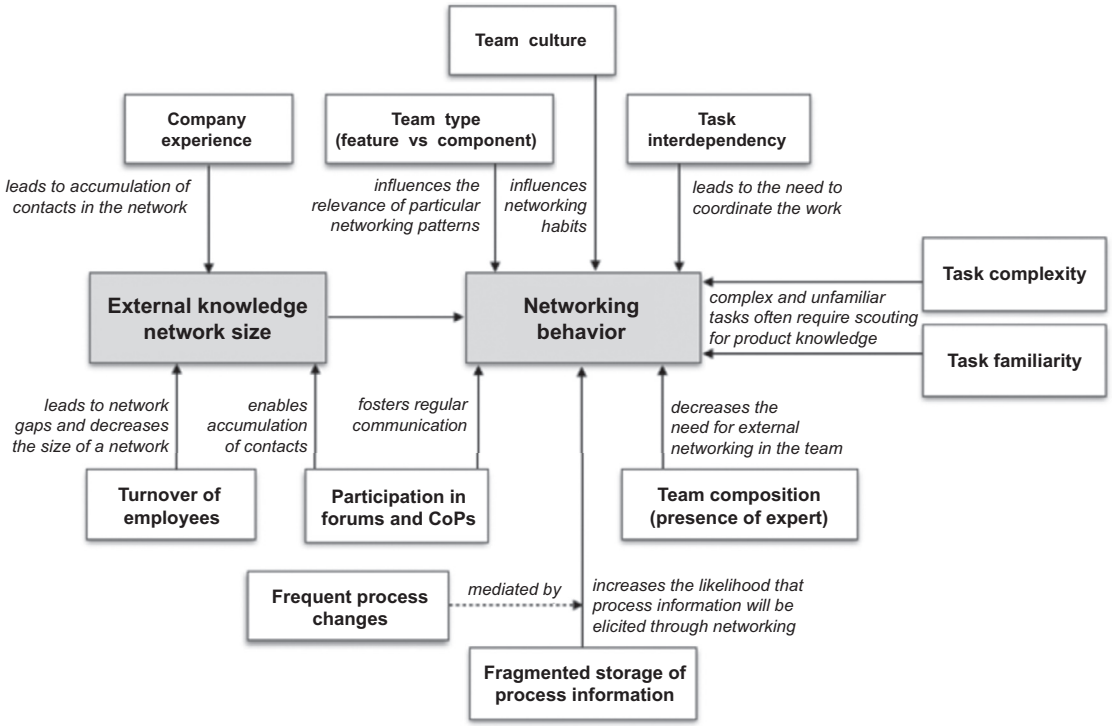

Fig. 4. Factors that influence team external-knowledge networks and networking behavior.

nical experts are. Our findings are consonant with the findings of Paasivaara and Lassenius [17] in that CoPs in large-scale developments can be used for knowledge sharing and coordination purposes.

In the following, we describe factors that motivate particular networking behavior of software teams:

- ***Familiar and less complex tasks*** reduce the need for the networking associated with scouting for product knowledge. In seven out of ten teams, individual team members solved the tasks using their own expertise or by accessing the source code and documentation. This is in line with existing research suggesting that rich external knowledge networks are more valuable when teams are working on interdependent or unfamiliar tasks [27].
- In contrast, ***unfamiliar tasks*** and ***lack of product knowledge in the team*** forces team members to reach out and scout for knowledge. This is related to the knowledge of the task at hand (e.g., a specific feature), and the ***knowledge of dependencies*** (e.g., a related component). Therefore, we found that networking is important both for new teams and mature teams challenged by unfamiliar or complex inter-dependent tasks. The size of the network, as noted earlier, will be larger for teams that have long company experience and who know where to scout for knowledge, whereas teams with little company experience will depend on the few experts in their network.
- ***Experts coordinate a large amount of knowledge*** into the teams and between the teams. Our findings are consonant with the findings of Ovaska et al. [19] on large-scale projects who found that a 'chief architect' was required to communicate the system's structures and solutions to achieve a common understanding of the system architecture and to help in coordinating development work.
  Formal experts also fulfill the role of boundary-spanners [16,29,30], which help in situations of poor cross-site awareness [40,41]. Simultaneously, teams may over-rely on the experts. This can lead to what are called 'structural holes' [26], meaning that developers from different teams circulate in different information flows with only experts bridging them, as was also found in related empirical studies on global software engineering [30].
- ***Fragmented storage of process information*** mediated by ***frequent process changes*** forces teams to seek updates and exchange process-related know-how, which is often facilitated in disciplined forums and personal cross-team connections. We identified this networking pattern in Ericsson, while teams in ABB, where working methods are more stable and standardized, did not report this type of knowledge exchange.
- A ***type of teams*** (feature teams versus component teams) determined team engagement in certain knowledge-sharing patterns. Feature teams scouted for knowledge from formal and informal experts to obtain access to the product knowledge needed for their feature development and particularly for unfamiliar tasks, while disciplined component teams coordinated their work tasks to be able to satisfy the needs of component interaction including familiar tasks.
- ***Culture*** played an important role in the networking behavior of the teams. From the interviews in Sweden, we found support for a belief that ***national culture*** is one reason for differences in networking behavior; for example, Chinese developers were assumed to be shy about external networking. However, our findings suggest that personal differences and the ***team culture*** have a stronger influence on the networking behavior. We found teams with small and large networks, and individuals with a very large number of contacts versus those with only a few, in both companies and in all locations. The influence of team culture was also evident in the team responses regarding their approach to problem solving. Evidently, there are differences in the role of networking and teamwork (social capital) even among the Swedish teams in both companies, with respect to the network size and dominance of networking patterns.

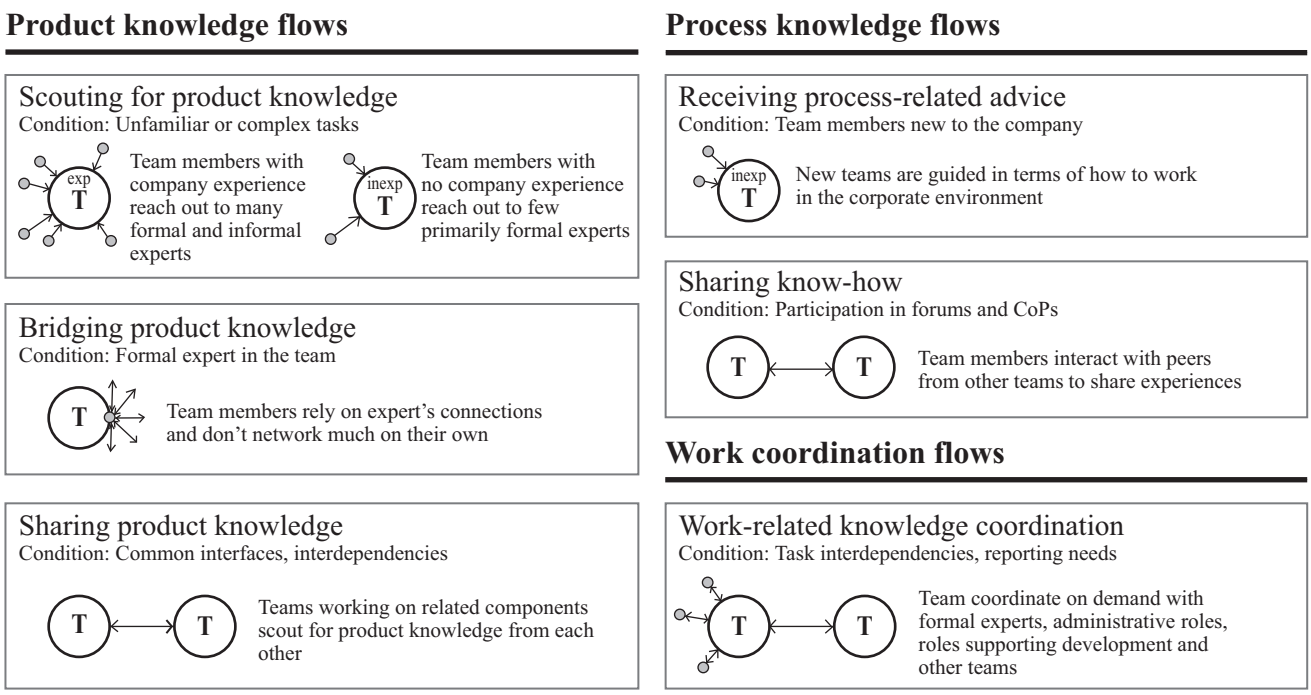

Fig. 5. Summary of identified networking patterns.

Additionally, we found that organizational infrastructure might have a secondary effect on the networking behavior. Our findings confirmed that practices that are a part of a so-called spatial school of knowledge management, i.e., practices focusing on designing office space to foster knowledge sharing, receive too little attention [42]. For cross-site interaction, we believe that companies should follow the advice of Licorish and MacDonell [13] who emphasized the importance of availability of adequate communication channels that help maintain a positive working climate, and therefore address the negative effects of distance.

All these factors are summarized in Fig. 5, which gives an overview of the influence on team external-knowledge networks and networking behavior. In summary, long company experience and attending CoPs have a positive impact on the network size, while turnover reduces it. Whether the potential in the network is used depends on how often people meet in formal arenas (e.g., CoPs), the type of work they are doing (task profile), task complexity, task familiarity, how interdependent the tasks are, team culture, team composition (experts in the team), and how process information is stored.

5.2. Networking patterns

Based on our analysis of the knowledge flows within each case (see the survey results presented in Tables 5 and 8 in Section 4), we identified a number of important networking patterns, which are presented in Sections 4.2 and 4.3 and summarized in Fig. 5. Knowledge flows reported by the teams qualified as a pattern if the flows were regular and common for one or more teams. The identified six networking patterns related to the product, the process, and work coordination form the basis of the networking behavior of software teams in the context of large-scale projects.

The main six networking patterns are summarized and illustrated in Fig. 5. In the figure we visualize a team or two teams (T in the circle), relative number of contacts in the network (many or few), and knowledge flow directions (incoming, outgoing or exchange). Each pattern is given a label which determines the purpose of knowledge acquisition, sharing or exchange.

We identified three patterns for the product-knowledge flows. The first pattern, Scouting for product knowledge, with two contexts (for team members with company experience and for team members with no company experience) was supported by both cases. The second pattern, Bridging product knowledge, emerged in the Ericsson case, while the third pattern, Sharing product knowledge, was the most dominant networking pattern in the ABB case. The two process knowledge flows, Receiving process-related advice and Sharing know-how, were found only in the Ericsson case. Finally, the Work-related knowledge coordination pattern was common for both cases.

By measuring the interaction frequency and distinguishing incoming, outgoing, and exchange flows, our findings complement related studies, e.g., the one by Licorish and MacDonell [13], who classified the identified knowledge-sharing patterns in terms of questions, answers, discussions, comments, reflections, scaffolding, instructions, and gratitude, and evaluated the dominance of each pattern during the execution of a project. In addition, we studied different teams in different circumstances, and came up with a number of common patterns reflecting the networking needs of the teams.

5.3. Limitations

The study has some obvious limitations in the reliability and generalizability of our findings.

The reliability of our findings is affected by a number of factors related to how we collected the survey data and constructed the knowledge networks. The results of the survey depend on how each actor conceptualized the meaning of the questions in the survey, and then selected the actors in their social networks as those belonging to their "knowledge network." To address this threat, we gave identical instructions to all teams participating in our research, and opportunities for clarification, if needed. Furthermore, the data related to knowledge sharing elicited through the survey was self-reported and thus represent the perception of the network and not an actual complete social network of individuals in the studied teams (informant bias [43]). Consequently, respon-

dents could have exaggerated or underestimated their knowledge and information-sharing connections. This and other related validity concerns are often addressed by employing a test-retest method to evaluate informant reliability [43]. However, we were unable to do so because of the limited availability of respondents for the researchers.

It is important to note that while we studied the network characteristics, we did not study the effect of human capital on the network. Human capital in the form of knowledge and skills embedded in the employees is expected to provide positive contributions to the social capital. Through human capital, a company will be able to effectively increase the commitments and reciprocal interactions within the network; thus, leading to a higher level of social capital related to interrelations, i.e., the greater the human capital is, the greater is the social capital [44].

The relationship model of factors and their influence on the external network size and networking behavior also has a number of limitations. It is based on only two projects in two companies, and a diverse sample of teams in each project. Although it helped to broaden the scope of our investigation, it also reduced the validity of our findings, e.g., false attribution of networking behavior to particular team-profile characteristics. To address these validity threats, we have included only factors supported by multiple sources of evidence and used multiple researchers to enhance the creative potential and the confidence in the findings [32]. We acknowledge that more evidence is needed to validate and enhance the meaning of the relationships in the model.

Finally, the generalizability of our work is influenced by our sampling strategy and the cases included in our investigation. We only studied team knowledge networks in two large-scale development environments in two companies. Although we employed a number of instruments to improve the reliability of our conclusions (e.g., data, methodological, and cross-case triangulation), they might reflect the peculiarities of the selected cases. Notably, our conclusions target large companies working on multi-team distributed projects, developing systems that evolve over a long period of time, and the applicability of our findings for other projects, e.g., green-field development, is unknown.

6. Conclusions

In this paper, we shared the findings from a descriptive study on the role of social capital, knowledge networks, and networking in two distributed large-scale development environments. In response to our research question (*What influences team knowledge networks and networking behavior in large-scale projects?),* we identified a number of factors that affected the teams' external knowledge networks and factors that determined networking behavior, and described the mechanisms that enabled and facilitated access to resources within a large-scale project. Our findings suggested that networking behavior and characteristics of knowledge networks vary even within the same company and the same site.

Despite the differences between the two companies, i.e., the team type (cross-functional self-managing feature teams versus disciplined component teams), we found many similarities in terms of the size of the teams' external networks and the factors influencing the networks and networking behavior. Furthermore, we learned that networking and access to experts in a team's knowledge network were crucial for new teams and teams working on new, unfamiliar, complex, or interdependent tasks. The importance of the network was particularly evident in large-scale projects, in which the accumulation of knowledge takes a long time and specialized knowledge is distributed across different units within the development organization. Finally, from our findings, we concluded that networking is the primary mechanism for solving complex tasks independent of a team's intellectual capital profile in large-scale projects.

One important implication of our findings is that software companies should explicitly cultivate networking. We suggest that strengthening the accumulation of social capital of all software teams in large-scale projects is at least as important as the development of the human and organizational capital. At the very least, this study points to the need to support newly created teams in their attempt to figure out who is who, and who knows what, both within the team and in the larger project network.

Acknowledgment

We are sincerely thankful to all participants involved in our study for their interest and feedback. We particularly thank the reviewers of this article for their thorough reviews and constructive feedback that has helped us greatly. This research is funded by the Swedish Knowledge Foundation under the KK-Hög grant 2012/0200, and by the Smiglo project, partly funded by the Research Council of Norway under grant 235359/O30.

Appendix. Survey questions

Open question
I exchange knowledge with (Name, Surname)
[Awareness]
strongly disagree, neutral, agree, strongly agree
I am aware of that person's area of expertise
I am aware of the knowledge that person needs for his/her job
[Availability]
strongly disagree, neutral, agree, strongly agree
It's easy for me to access that person
[Knowledge transfer, Frequency]
never, rarely, sometimes, often, daily
How often do you transfer knowledge to that person?
[Knowledge transfer, Content]
open question
What kind of knowledge do you transfer to that person?
[Knowledge retrieval, Frequency]
never, rarely, sometimes, often, daily
How often do you receive knowledge from that person?
[Knowledge retrieval: Content]
open question
What kind of knowledge do you receive from that person?

References

[1] K.C.D.K.L. Smith, The Perils of Petascale IT Projects, FCW: The Business of Federal Technology, 2014.
[2] B. Flyvbjerg, A. Budzier, Why your IT project may be riskier than you think, Harvard Bus. Rev. 89: (2011) 23–25.
[3] B. Flyvbjerg, What you should know about megaprojects and why: an overview, Project Manage. J. 45 (2) (2014) 6–19.
[4] B.S. Blichfeldt, P. Eskerod, Project portfolio management – there's more to it than what management enacts, Int. J. Project Manage. 26 (4) (2008) 357–365.
[5] V.T. Rajlich, K.H. Bennett, A staged model for the software life cycle, Computer 33 (7) (2000) 66–71.
[6] R.S. Burt, The network structure of social capital, Res. Organ. Behav. 22 (2000) 345–423.
[7] J. Nahapiet, S. Ghoshal, Social capital, intellectual capital, and the organizational advantage, Acad. Manage. Rev. 23 (2) (1998) 242–266.
[8] H.-D. Yang, H.-R. Kang, R.M. Mason, An exploratory study on meta skills in software development teams: antecedent cooperation skills and personality for shared mental models, Eur. J. Inf. Syst. 17 (1) (2008) 47–61.
[9] T.W. Malone, K. Crowston, The interdisciplinary study of coordination, ACM Comput. Surv. 26 (1) (1994) 87–119.
[10] G.A. Okhuysen, B.A. Bechky, Coordination in organizations: an integrative perspective, Acad. Manage. Ann. 3 (2009) 463–502.
[11] R.E. Kraut, L.A. Streeter, Coordination in software development, Commun. ACM 38 (3) (1995) 69–81.
[12] M.H. Fagan, The influence of creative style and climate on software development team creativity: an exploratory study, J. Comput. Inf. Syst. 44 (3) (2004) 73–80.
[13] S.A. Licorish, S.G. MacDonell, Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: a longitudinal study, Inf. Softw. Technol. 56 (12) (2014) 1578–1596.

[14] N. Lundberg, H. Tellioğlu, Understanding complex coordination processes in health care, Scand. J. Inf. Syst. 11 (2) (1999) 157–181.
[15] D.E. Strode, S.L. Huff, B. Hope, S. Link, Coordination in co-located agile software development projects, J. Syst. Softw. 85 (6) (2012) 1222–1238.
[16] A. Boden, G. Avram, L. Bannon, V. Wulf, Knowledge management in distributed software development teams-does culture matter? Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on, IEEE, 2009.
[17] M. Paasivaara, C. Lassenius, Communities of practice in a large distributed agile software development organization – Case Ericsson, Inf. Softw. Technol. 56 (12) (2014) 1556–1577.
[18] E. Wenger, Communities of Practise : Learning, Meaning and Identity, Cambridge University Press, Cambridge, UK, 1998.
[19] P. Ovaska, M. Rossi, P. Marttiin, Architecture as a coordination tool in multi-site software development, Softw. Process 8 (4) (2003) 233–247.
[20] J.D. Herbsleb, R.E. Grinter, Architectures, coordination, and distance: Conway's law and beyond, Software, IEEE 16 (5) (1999) 63–70.
[21] M. Paasivaara, C. Lassenius, V.T. Heikkila, Inter-team coordination in large-scale globally distributed scrum: do Scrum-of-Scrums really work? Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on, 2012.
[22] P. Bourdieu, The forms of capital, Handbook of Theory and Research for the Sociology of Education, J. Richardson, New York, Greenwood, 1986.
[23] G.R. Oliver, J. D'Ambra, C. Van Toorn, in: Evaluating an Approach to Sharing Software Engineering Knowledge to Facilitate Learning, Managing Software Engineering Knowledge, Springer, 2003, pp. 119–134.
[24] J. Mathieu, M.T. Maynard, T. Rapp, L. Gilson, Team effectiveness 1997-2007: a review of recent advancements and a glimpse into the future, J. Manage. 34 (3) (2008) 410–476.
[25] C. Wohlin, D. Šmite, N.B. Moe, A general theory of software engineering: balancing human, social and organizational capitals, J. Syst. Softw. 109 (2015) 229–242.
[26] R.S. Burt, Structural Holes: The Social Structure of Competition, Harvard University Press, Cambridge, MA, 1992.
[27] K. Lewis, Measuring transactive memory systems in the field: scale development and validation, J. Appl. Psychol. 88 (4) (2003) 587.
[28] H. Mintzberg, Mintzberg on Management: Inside Our Strange World of Organizations, Free Press, New York, 1989.
[29] A. Johri, Boundary spanning knowledge broker: an emerging role in global engineering firms, Frontiers in Education Conference, 2008. FIE 2008. 38th Annual, IEEE, 2008.
[30] C. Manteli, B. Van Den Hooff, H. Van Vliet, The effect of governance on global software development: an empirical research in transactive memory systems, Inf. Softw. Technol. 56 (10) (2014) 1309–1321.
[31] P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, John Wiley & Sons, 2012.
[32] K.M. Eisenhardt, Building theories from case study research, Acad. Manage. Rev. 14 (4) (1989) 532–550.
[33] R.K. Yin, Case Study Research: Design and Methods, Sage, Thousand Oaks, Calif., 2008.
[34] D.W. Stewart, P.N. Shamdasani, Focus Groups: Theory and Practice, Sage Publications, 2014.
[35] C. Robson, Real World Research, Blackwell, Oxford, 2002.
[36] C.B. Seaman, Qualitative methods in empirical studies of software engineering, IEEE Trans. Softw. Eng. 25 (4) (1999) 557–572.
[37] S. Faraj, L. Sproull, Coordinating expertise in software development teams, Manage. Sci. 46 (12) (2000) 1554–1568.
[38] S. Beecham, N. Baddoo, T. Hall, H. Robinson, H. Sharp, Motivation in software engineering: a systematic literature review, Inf. Softw. Technol. 50 (9–10) (2008) 860–878.
[39] I. Rus, M. Lindvall, Knowledge management in software engineering, Software, IEEE 19 (3) (2002) 26–38.
[40] C. Gutwin, K. Schneider, D. Paquette, R. Penner, Supporting group awareness in distributed software development, in: Engineering Human Computer Interaction and Interactive Systems, Springer, 2004, pp. 383–397.
[41] Z.U.R. Kiani, D. Mite, A. Riaz, Measuring awareness in cross-team collaborations–distance matters, Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on, IEEE, 2013.
[42] T. Dingsoyr, F.O. Bjornson, F. Shull, What do we know about knowledge management? Practical implications for software engineering, Software, IEEE 26 (3) (2009) 100–103.
[43] D. Knoke, S. Yang, Social Network Analysis, Sage, 2008.
[44] C.J. Chen, H.A. Shih, S.Y. Yang, The role of intellectual capital in knowledge transfer, IEEE Trans. Eng. Manage. 56 (3) (2009) 402–411.