The Role of Domain Knowledge and Cross-Functional Communication in Socio-Technical Coordination

Daniela Damian[1], Remko Helms[2], Irwin Kwan[3], Sabrina Marczak[4], and Benjamin Koelewijn[2]

[1]Dept. of Computer Science, University of Victoria, Victoria, Canada, danielad@cs.uvic.ca

[2]Dept. of Computer Science, Utrecht University, Utrecht, The Netherlands, r.w.helms@uu.nl, bskoelewijn@gmail.com

[3]School of Elec. Engr. and Computer Science, Oregon State University, Corvallis, USA, kwan@eecs.oregonstate.edu

[4]Computer Science School, PUCRS, Porto Alegre, Brazil, sabrina.marczak@pucrs.br

***Abstract*—Software projects involve diverse roles and artifacts that have dependencies to requirements. Project team members in different roles need to coordinate but their coordination is affected by the availability of domain knowledge, which is distributed among different project members, and organizational structures that control cross-functional communication. Our study examines how information flowed between different roles in two software projects that had contrasting distributions of domain knowledge and different communication structures. Using observations, interviews, and surveys, we examined how diverse roles working on requirements and their related artifacts coordinated along task dependencies. We found that communication only partially matched task dependencies and that team members that are boundary spanners have extensive domain knowledge and hold key positions in the control structure. These findings have implications on how organizational structures interfere with task assignments and influence communication in the project, suggesting how practitioners can adjust team configuration and communication structures.**

***Index Terms*—Software coordination, cross-functional communication, global software teams, socio-technical coordination, domain knowledge, distributed development**

I. INTRODUCTION

When project members with interdependent tasks do not communicate effectively, coordination breakdowns occur and result in integration failures [1], lower developer productivity [2][3] and defects [4]. Studies that replicated these findings are based largely on developers' work on code artifacts (e.g. [2]), but coordination across a project involves a wider set of artifacts and functional roles other than developers. For example, changes in requirements may trigger negotiations between project managers and customers, changes in architecture and the associated code, or revising testing procedures by the quality assurance staff.

In software engineering research, we have limited knowledge about the influence of the thin distribution of domain knowledge on the coordination of various roles along task dependencies and across team boundaries. Various project roles possess differing levels of domain knowledge—knowledge about the users' needs, their business and system environment that is relevant to their tasks [5]. Managers have to make tradeoffs in task assignments by balancing both a person's application domain and technical knowledge [6] and often mandate cross-functional communication by assigning certain roles to liaise between functional or geographically remote teams [7][8][9]. While such organizational structures generally aim to increase the efficiency of communication in the project, they influence software projects in various ways [9][10][11][12].

In this paper, we present an exploratory study in which we investigated the influence that the cross-functional communication structure and the distribution of domain knowledge (referred to as organizational structures henceforth) had on coordination in a software team. We examine and contrast two projects from a large IT organization that had different distributions of application domain knowledge and different cross-functional communication structures that controlled information flow between roles. We specifically investigate how diverse roles working on requirements and their related downstream artifacts coordinate along the dependencies among their tasks. To uncover details of coordination in these projects, we use a case study method to analyze contextual information about tasks, cross-functional communication, and the distribution of domain knowledge.

Our analyses revealed that in order to understand coordination one has to look beyond task dependencies in the project. We found that the various roles in coordination engage in more communication than anticipated from task dependencies, which could be explained by adherence to the cross-functional communication structure as well as patterns of seeking domain knowledge from certain roles. In particular, we find that team members tend to communicate across applications within the same domain, that members engage in backchannel communication to complete their tasks, and that members brokering communication across application domains have extensive domain knowledge.

The contributions of this paper include (1) the empirical evidence that coordination is affected by the interplay of organizational structures and task dependencies and (2) the strategies employed to overcome the thin spread of domain knowledge in the projects we studied. Our findings have implications with respect to the study of socio-technical

coordination to account for these additional organizational structures, and configuring teams to optimize the dissemination of domain knowledge in software projects.

II. Related Work and Research Questions

Research into software team coordination has confirmed that aligning organizational factors and technical factors affects software quality and cost [1][2][13][14]. In particular, these researchers have analyzed *socio-technical alignment*, which examines how the technical aspects of software engineering are matched by the work-related interactions of software developers. One measure of socio-technical alignment is *socio-technical congruence* (STC) [2], which calculates the alignment of *actual communication* with the anticipated *coordination needs* of team members based on technical task assignments and task dependencies. Applications of this approach yielded significant empirical evidence about coordination in software development—higher socio-technical congruence generally correlates with higher developer productivity [2] and reduced integration failures [1]. Managers and researchers could use STC to diagnose and improve team coordination [13].

However, most research on socio-technical alignment has been focused on developers' work from repositories and development artifacts such as code [4], defects [15] or software builds [1]; studies of coordination in open-source tend to focus on developers as well (e.g. [16]). This focus on code and defects overlooks the coordination that takes place within a wider set of stakeholders such as business analysts, testers, requirements analysts and project managers. A way to study coordination between diverse roles is to look at a higher level of abstraction: *requirements*.

Requirements are central elements in project planning and resourcing and provide a focal point around which team members coordinate their tasks [15][17][18]. Requirements create dependencies among downstream artifacts such as design, architecture, code, and test cases. Environments (e.g. [19]) now leverage traceability links between requirements and these associated artifacts to enable collaboration and software governance throughout the entire project life cycle. Recent studies found that communication driven by requirements takes place within cross-functional teams involving developers, business analysts and testers [17][20] and that the most predominant reason for communication was changes in requirements [21].

Despite this evidence, we have limited knowledge as to whether the communication of these multiple roles aligns with their task dependencies and consequently about the role of task assignments in coordination. To complete their tasks, team members often stretch their communication by reaching out to those that possess in-depth domain knowledge across different teams and geographical locations [7]. Therefore, our first research question is exploratory and seeks to unveil details of the elements of the socio-technical alignment of software teams, namely the task dependencies as well as the communication of the wider set of stakeholders:

RQ1: What is the nature of task dependencies, project communication and socio-technical alignment in a requirements view on coordination?

Organizational structures that control cross-functional communication is one factor that may influence the communication among the various stakeholders. Such a structure typically introduces a hierarchy in which certain organizational roles control information flow within and across teams or departments [25]. These hierarchies and groups exist in all kinds of project sizes [16]. Bird et al. [10] identified that in popular open-source software projects, communication was centered on small, interoperating groups of developers, and Hinds and McGrath [9] observed that dense structures tended to be associated with communication problems. In contrast, Cataldo and Ehrlich found that hierarchical networks delivered more features than close-knit networks but had lower quality [11]. Social network analysis has also been used to identify a relationship between communication structure and software quality [1][14]. Although organizational structures aim to prevent communication issues and increase communication efficiency [25], they may also impede knowledge flow if the structure is not defined according to task assignments, and studies of organizational behavior document the role of informal networks in task accomplishment (e.g. [27][28]). This leads us to our second research question:

RQ2: How does the cross-functional communication structure influence actual communication of various roles in coordination?

Involving multiple functional roles also introduces differences in *domain knowledge* within the team. Domain knowledge is the implicit knowledge about client needs, their business domain and the system's environment [29]. Communicating domain knowledge to others improves team's understanding and can increase team buy-in [30]. The domain expert often brokers knowledge across geographical boundaries [8] or roles [31], bridges gaps between people that are otherwise not communicating [7][32][31], and is able to promote innovation [22][33]. Often, the domain expert does not plan to be in this brokering position, which leads to limited access to that expert [6][30]. Thus, coordination and communication patterns within projects may not align with task coordination and instead follow informal connections governed by domain expertise [27][28]. A requirements perspective on coordination allows us to examine communication driven by the distribution of domain knowledge in the project.

RQ3: How does the spread of domain knowledge influence communication of various roles in coordination?

III. RESEARCH METHODOLOGY

We used a multiple case study methodology [34] and a combination of quantitative and qualitative data collection methods to study coordination in two projects called SHIP and APP within a large multinational organization that we call ORG. ORG develops software applications to support its primary business of selling and supporting IT equipment. ORG releases quarterly versions of its software applications, which are organized in portfolios according to the ORG business area they serve. ORG's headquarters are in the United States, while its development centers are located in the United States, Brazil, India, and Russia.

SHIP and APP serve different business areas within ORG and, at the time of our study, had different team configurations in the US and the offshore locations, different distributions of application domain knowledge among its members, as well as different cross-functional communication structures to cope with its newly formed relationship with Brazil and India. Both project teams were considered successful, delivering on time and on-budget.

A. Project Descriptions

The SHIP project enhanced and maintained a critical internal software application supporting ORG's shipping process. The application was eight years old and was outsourced to Brazil three years prior to our study. We investigated a four-month release during which the application was updated to accommodate changes in ORG's business process and the database infrastructure.

Project team configuration. The project team (14 persons) consisted of its original four members in US: system architect, developer leader, and 2 developers, and 10 newly hired members in Brazil: developer leader, test leader, 5 developers and 3 testers. In Brazil, the team was distributed across three buildings on-site. Additionally there were 3 business partners located in the United States who acted as customers for the application. They were representatives of ORG's shipping production team (e.g. logistic manager and environment coordinator) who worked together with the ORG IT infrastructure team.

Knowledge about the application and its domain. This team had access to up-to-date project documentation detailing its requirements and architecture. The team could also access the application's domain knowledge through the American developer leader, system architect and a senior developer originally involved in the application's inception. New Brazilian hires travelled to the United States development site for six months for training. Developer leaders were in charge of gathering and negotiating requirements for new releases with the internal customers. The project manager's responsibility was to manage the schedule, resources, and budget. The system architect supported the developer leaders regarding architecture, though this did not often occur in this iteration due to the few architecture changes.

Cross-functional communication structure. SHIP's communication structure is illustrated in Fig. 1b (legend for all our networks is in Fig. 1a). The symbols represent team roles; the connecting dotted lines represent the project's hierarchical control structure, whereby some roles need to contact the 'next in line' role for communication with the others. For example, the developers and testers were asked to contact their respective leaders for any communication with the project manager or system architect, though they were encouraged to communicate directly if necessary.

TABLE I. ANONYMIZED SAMPLE REQUIREMENTS

SHIP project	APP project
Change shipping label to a new standard	Add a filter option for the *production issues list* to facilitate searching for a specific issue
Allow email notification when order has been shipped or is available for pickup	Allow users to schedule an automated distribution of the selected production report by e-mail
Improve the calculation of the estimated delivery date to improve customer satisfaction.	Build a new configurable notification feature to replace the current warning mechanism that notifies workers that a part has to be returned to the physical inventory

The APP project contained around one hundred applications within four application portfolios. Though the applications were developed originally in the United States, the current release was outsourced to a team in Brazil and three testers in India. We examined a subset of the APP requirements.

Project team configuration. Twenty-five project members were assigned to the requirements examined in our study, though the entire project had 45 members. All 25 were located in Brazil: 4 requirements analysts (RAs), 1 test leader, 7 testers, 3 developer leaders (dev leads), and 10 developers. About a third were contractors or new hires in the company. The team members worked in three buildings on-site. There were also five business partners in the United States who were internal clients for these applications.

Knowledge about the applications and their domain. In contrast to SHIP, the distribution of application domain knowledge among the project members in APP was uneven. The domains of the four application portfolios were: ORG employee career tracking and advancement, sales reports for different world regions, incident compliance in ORG's manufacturing plants, and project management. The new team in Brazil had no previous experience with the application portfolios and could not contact the original designers. There was no documentation on the applications' functionality or context of use within ORG. Although the Brazilian team members were recruited based on their knowledge of the four application portfolios, the developers and their leaders had to reverse-engineer the applications over the course of four months with help from end users based in the United States to understand their intended functionality. For the new releases,

TABLE II. REQ. MAP TO APPLICATION PORTFOLIOS IN APP PROJECT

Appl. Porfolio	A	A	A	A	B	B	B	C	C	D
Application ID	1	2	3	4	5	6	7	8	9	10
Num. Req.	1	1	1	1	4	1	2	3	2	4

the requirements analysts and dev leads gathered and negotiated requirements with the United States customers, and prioritized them together with the project manager.

Cross-functional communication structure. Because this team consisted mainly of novice developers and testers in Brazil, the United States headquarters imposed a restrictive communication structure (Fig. 1c). The dev and test leaders were the points of contact between the teams and the RAs, and no direct communication between the different roles was supported.

B. Data Collection and Conceptualizations

Two of the authors conducted a three-month on-site observation of both project teams, inspected project requirements and planning documentation, and deployed a survey. They also attended project meetings, shadowed project members, and interviewed team members to validate our understanding. We used social network analysis [35] to represent and analyze data about the project members and their coordination.

Requirements and their dependencies. We were given access to study 18 requirements in the SHIP project and 20 requirements in the APP project (see examples in Table I). There were 5 sets and 4 sets of requirements' dependencies (i.e. "refined-to", "requires", and "conflicts-with" dependencies) in the SHIP and the APP project respectively. The 20 requirements in the APP project belonged to 10 applications in 4 application portfolios as shown in Table II.

Task dependencies and anticipated coordination needs. We used the project planning documentation to identify project members' assignment to the tasks of analysis, design, coding and testing of each requirement. We also identified interdependent tasks in consultation with the development and test leaders. We validated this information using detailed meetings with the design team, particularly with the team leaders. In a requirements view on coordination, two project members have a task dependency and thus an anticipated *coordination need* if they work on tasks related to a set of interdependent requirements. We represent these task dependencies in a coordination needs network (see each projects' networks in Fig. 2a and 2b).

Actual communication. To capture communication about tasks that specifically related to work on requirements and their interdependent artifacts, we deployed a survey 3 months into the project, at the end of development phase. We used a survey because surveys are a standard procedure in social network analysis research and the survey covered a variety of communication channels (face-to-face, telephone, email) available to the distributed teams. The survey was customized for each project member. It presented a list of members with which the respondent had a coordination need (based on planning documentation) and asked them to indicate who they communicated with in their tasks, and which requirement that was related to. The respondent could also indicate additional members s/he communicated with. We also asked for the reason of communication, which was one of the following: requirements clarification, negotiation, communication of requirements changes, or coordination of activities. Our survey is available from http://bit.ly/NJXBRB. As much as we could, we validated the information collected through our on-site observations and interviews.

TABLE III. ATTRIBUTES OF SHIP AND APP

Attribute	APP project	SHIP project
Nodes (members in actual communication)	**35**	**20**
Communication links	**104**	**103**
Anticipated coordination needs	**112**	**126**
Socio-technical congruence index	**0.58**	**0.64**
Emergent nodes	**10** (/35= 29%)	**6** (/20= 30%)
Interactions involving emergent nodes	**11** (of 104= 11%)	**14**(/103= 14%)
Emergent interactions	**39** (/104= 38%)	**22**(/103= 21%)
Emergent interactions in line with control hierarchy	**29** (/39= 74%)	**16** (/22= 72%)
Emergent interactions within same application portfolio	**24** (/39= 62%)	**22** (/22=100%)
Backchannel communication	**55** (/104 = 53%)	**26** (/103=25%)

IV. ANALYSIS AND RESULTS

RQ1: What is the nature of task dependencies, project communication, and socio-technical alignment in a requirements view on coordination?

Table III presents summary data for each of the projects. To explore socio-technical alignment in interactions involving multiple functional roles, we first computed the socio-technical congruence index by Cataldo and colleagues [2] and which calculates the percentage of the anticipated coordination needs that are satisfied by actual communication taking place during the project. We found only 64% and 58% match in the two projects respectively, implying that at least one third of the anticipated coordination needs between team members did not have corresponding communication. However, both projects had substantial *emergent communication*, which are communication links not predicted by coordination needs: 38% and 21% in APP and SHIP projects respectively. We thus explored the nature of both task dependencies and actual communication and present the findings below.

Task dependencies are grouped within application portfolios. To explore task dependencies in the two projects, we constructed the *coordination needs network* inferred from these dependencies (Fig. 2a and b). To visualize and analyze the networks in this study we used the social network analysis tool Netminer 4.

To compare the properties of networks we laid out the graphs using the Spring algorithm [36] that minimizes edge overlap. In the coordination needs network, two project members are connected if their tasks relate to the same requirement or an interdependent set of requirements. The nodes' shape indicates a team member's role in the project while the size shows their experience in the organization (larger nodes are more experienced). The node color indicates location, where white indicates Brazil and darker color indicates the US. We also show members' assignment to the application or portfolios defined by ORG. While there is only

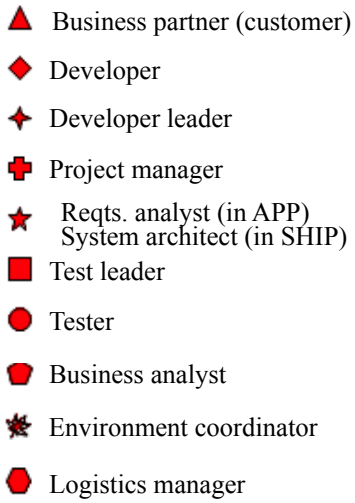

Fig. 1a. Legend

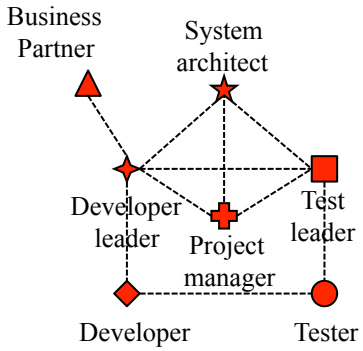

Fig. 1b. SHIP's hierarchical structure

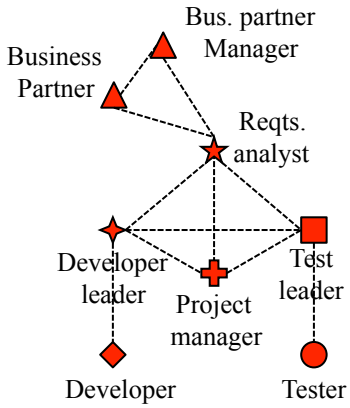

Fig. 1c. APP's hierarchical structure

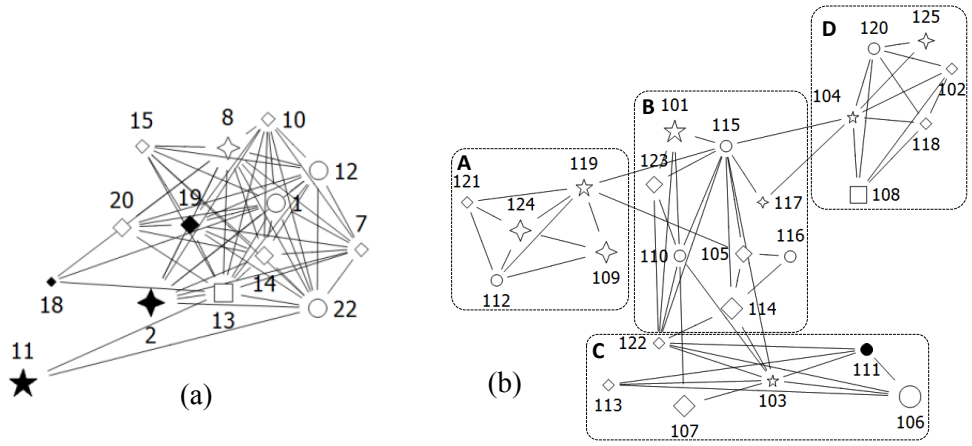

Fig. 2. **Coordination needs networks** in the SHIP (a) and APP project (b)

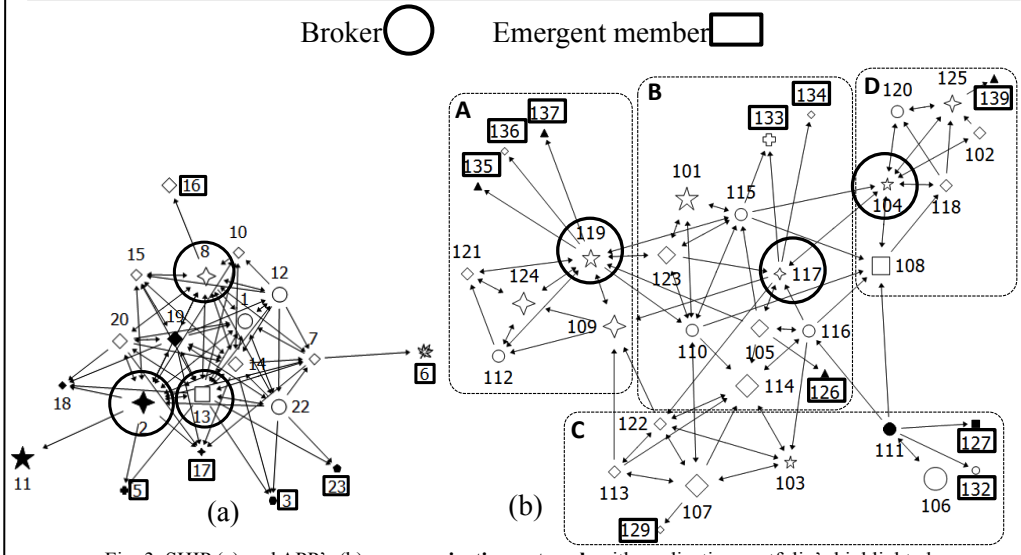

Fig. 3. SHIP (a) and APP's (b) **communication network** with application portfolio's highlighted (boxes)

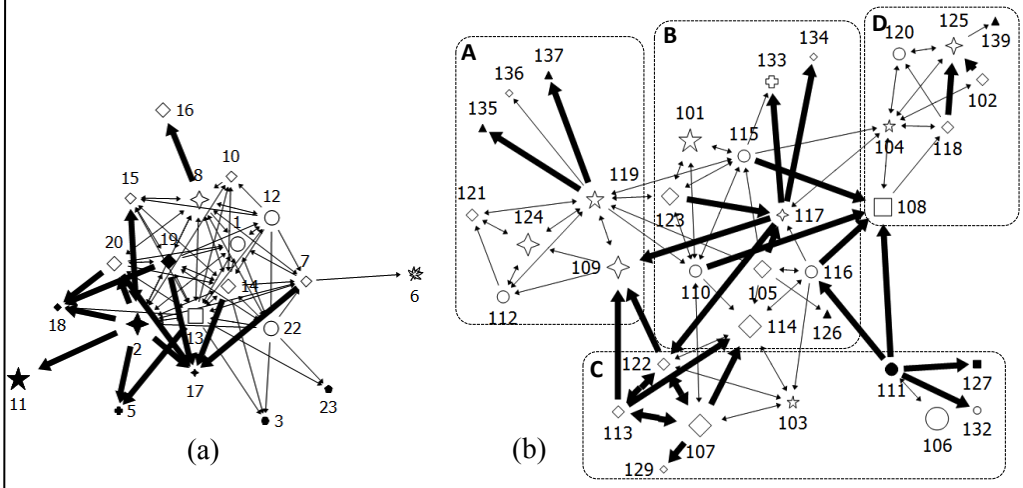

Fig. 4. Comm. links that do not follow task assignments but are **according to hierarchical structure** in SHIP (a) and APP project (b)

one application in SHIP, there are four application portfolios in APP (boxes A, B, C, D in Fig. 2b), for which project members have been assigned based on their knowledge of the application domains.

We found that the task dependencies within APP largely align with the grouping of applications into the four application portfolios (75% are within application portfolios). Since there were no architectural dependencies between the application portfolios, the few cross-portfolio dependencies were the result of some members being assigned to requirements in multiple portfolios due to their domain knowledge.

Actual communication is grouped within application portfolios. Fig. 3 illustrates the social networks constructed from the survey on actual communication in the two projects. The networks include the nodes in the coordination needs networks as well as additional members with which our respondents identified as having had communication. Two members are connected if they reported an instance of task communication. The node layout for the actual communication network is the same as for the coordination needs network to enable easy comparison of the networks.

We find that there are many more communication links in comparison to coordination needs and that communication largely fits within application portfolios (see Fig. 3b). To examine how tightly connected the portfolio-based groups are in APP relative to the entire project network we calculated the Segregation Matrix Index (SMI) [37] for each of these groups. The SMI index tests if a pre-defined group of nodes is segregated from the larger network and ranges from 0 (no segregation) to 1 (complete segregation). The results show high segregation (>0.6) in each application portfolio for both actual communication (Fig. 3b) and coordination needs (Fig. 2b) networks (Table IV). The high SMI index scores illustrate that team members working within the same applications portfolio tend to be more connected than team members working in different application portfolios.

TABLE IV. SMI ANALYSIS RESULTS FOR APP

Appl. Portfolios	SMI (communication)	SMI (coord. needs)
A	0.92	0.94
B	0.65	0.63
C	0.69	0.76
D	0.98	0.97

Emergent interactions involve team members that possess domain knowledge. Emergent communication occurred between team members who had no coordination needs. From Table III, 38% of the total communication is emergent in APP, involving 10 emergent members, while in SHIP 21% of communication is emergent, involving 6 emergent members. About 10% of the overall communication involved emergent members. One instance of emergent communication that we observed in SHIP is between the dev lead #8 (Fig. 3a) and developer #16 who was not assigned to work on any requirements. Dev lead #8 was working on the requirement to upgrade a particular component to a new technology and contacted developer #16 who had knowledge of the new technology in the customer's operational environment.

Table V lists the roles of the emergent communicators. Most of these members represent sources of application domain knowledge; they are customers, business analysts or environment coordinators, or had been involved in requirements negotiations in SHIP (e.g. dev leads) or in the reverse engineering efforts in APP (e.g. developers).

TABLE V. EMERGENT ROLES AND MEMBERS (NUMBERS INDICATE NODES ON THE SOCIAL NETWORK)

APP project	SHIP project
Bus. Partners/Customer (126, 135, 137, 139)	Developer (16)
Developer (129, 134, 136)	Developer Leader (17)
Test Leader (127)	Project Manager (5)
Tester (132)	Logistics manager/Cust (3)
Project Manager (133)	Environ. Coordinator (6)
	Business Analyst (23)

Communication brokers have domain knowledge. To identify communication brokers—those who mediate between members not communicating directly—we applied the Betweenness Centrality Index (BCI) [35], an index ranging from 0 to 1 (where 0 is low and 1 is high). We ranked the members' BCI score and considered a threshold calculated as the median + 1.5 * interquartile range [38] to identifying 'top' brokers from each project. Using this threshold three brokers stand out in each of the two projects. They are outlined in Table VI and marked by a circle in Fig. 3a and b.

In APP, the brokers are two RAs (#104 and #119) and a dev lead (#117). In SHIP, they are dev leads (#2 and #8), and test leader (#13). In both our projects, dev leads and RAs are brokers because of their knowledge of the application functionality and requirements. Our interviews with members playing these roles in both projects indicate that the RAs and dev leads were the points of contact when customers needed to be contacted.

To summarize, we observed that in these two projects, both the task dependencies and the actual communication were grouped by application portfolios, that there was more communication in the projects than anticipated by the task dependencies, and that the individuals that emerged as brokers in communication have extensive domain knowledge.

RQ2: How does the cross-functional communication structure influence actual communication of various roles in coordination?

To answer this question we analyzed whether actual communication instances matched the hierarchical control structure. We found that only 45% and 78% of the actual communication links were according to the mandated cross-functional communication structure in the APP and SHIP

TABLE VI. TOP BROKERS BY BETWEENNESS CENTRALITY INDEX (BCI)

APP Brokers by BCI		SHIP Brokers by BCI	
Req. Analyst #104	0.99	Dev lead #2	0.92
Req. Analyst #119	0.88	Dev lead #4	0.89
Dev lead #117	0.77	Test leader #13	0.82

project respectively. Because of the possibility that project members engaged in communication outside of the control structure in order to complete their tasks, we investigated the alignment between the actual communication and the cross-functional communication structure in light of the task assignments. We first examined whether any mismatch between the two was along a task dependency; then we examined to which extent the emergent communications—the communications not predicted by the task assignments—conformed to the cross-functional communication structure.

Task assignments create the need for backchannel communication. We found that 53% of communication in the APP project meets a task dependency but disregards the cross-functional communication structure. This communication is referred to as backchannel communication [28][20]. This is a high percentage for the APP project, though not surprising given that its mandated communication structure was restrictive of direct communication between developers and testers, as well as their communication with the RAs who were familiar with the requirements. An example situation that we observed included a tester who, suspecting that one of the defects he encountered relates to critical functionality, contacted the RA directly (the test leader he was supposed to contact was not available on that day) to confirm the criticality of the functionality. He was later praised for his proactive attitude as the defect was going to significantly delay an upcoming milestone. In contrast, the 25% backchannel communication in the SHIP project reflects its less restrictive cross-functional communication structure.

Most emergent communications align with mandated cross-functional communication structure. The emergent communications follow a different pattern: 72% and 74% of such communications respectively (see Table III) are in turn in line with the mandated structure. They are highlighted as bold lines in the networks in Fig. 4a and b. Our interviews and observations confirm this finding: emergent communications mostly involved contacting colleagues within the same application portfolio (8 links in APP and 2 in SHIP) or addressing team leaders for issue resolution (9 link in APP and 2 in SHIP). Situations we observed include a RA asking a customer for requirements clarifications, or a developer disagreeing with a change request for one of the requirements he was implementing and, after unsuccessfully advising with his fellow developers on the same requirement, deciding to talk to his dev lead – and in the same functional team -- to make a decision.

RQ3. How does the spread of domain knowledge influence communication of various roles in coordination?

To answer this question we examined to what extent the actual communication could be explained by the distribution of application domain knowledge in the two projects.

Communication clusters around application domains. The SMI analysis earlier (Table IV) showed that actual communication mainly takes place within the application portfolios, meaning that members communicate more within the same application portfolio than across portfolios. For SHIP this is evident as there is only one application, and for APP there are only 20 out of 104 links in between portfolios. Since there were no architectural dependencies between these applications, the underlying commonality among the applications within the same portfolio was the business domain that they were serving, suggesting that people within the same application portfolio were contacted for their application domain knowledge.

Our interviews confirmed that an underlying application domain knowledge network supported each application portfolio. All but one portfolio (Fig. 2b) were configured with a cross-functional team comprised of at least a developer, a tester, a dev lead and a RA. Among these roles, some were knowledgeable of the application domain through negotiating with the customer (e.g. RAs and dev leads) or through reverse engineering the application functionality (e.g. developers).

Most emergent communications seek domain knowledge within the same portfolio. Examining the emergent communications in detail shows that 62% emergent communications took place within the application portfolios of the APP project (thick lines in Fig. 5) and they were with members that represented sources of application domain knowledge. An example of emergent, domain knowledge-based communication between RAs and customers is shown in Fig. 5 and corroborated by the data we collected on reasons for communication: RA #119 asked for requirements clarifications from customers #135 and #137 in application portfolio A and dev lead #125 asked for clarifications and negotiated requirements with customer #139 in application portfolio D. We observed that in three of the four application portfolios there is at least one customer that emerged in the communications, indicating that project members sought customers for their domain knowledge.

Communication brokers are boundary spanners. To identify if communication brokers also mediated communication across multiple application domains in APP, we examined whether these brokers' communication across the boundaries of the four portfolios was higher than that of the non-brokers. We found that the three brokers had in

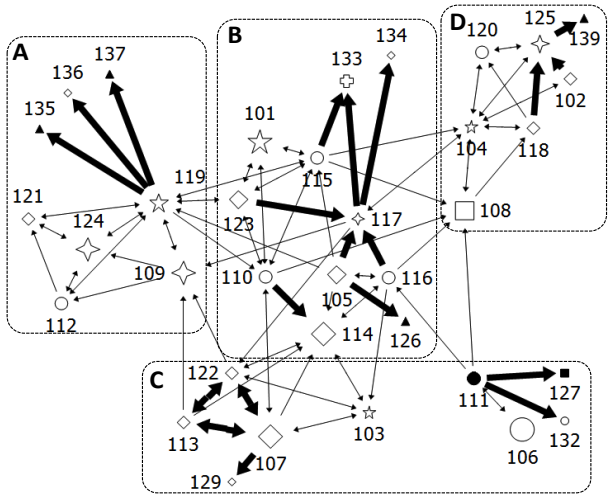

Fig. 5. APP's Communication that is emergent within same application portfolio

average 2.33 outbound communications, a number much higher than that of 0.63 communications for non-brokers.

This finding, combined with the fact that these brokers possessed domain knowledge suggests that brokers acted as boundary spanners [5] across the four application domains in APP. An alternate explanation to the brokers' communication across domains is that they had higher task dependencies than the other members (Fig. 2b). However, we observed that other members with high cross-domain task dependencies, such as #122, did not score high either as brokers or as boundary spanners, suggesting that higher task dependencies alone do not lead to boundary spanning.

To summarize the answers RQ2 and RQ3, we found that whenever the mandated communication structure did not align with the task dependencies members engaged in backchannel communication to complete their tasks, but most communications that emerged with members not in a task dependency was in adherence to the mandated communication structure. Domain knowledge was influential as well. Seeking application domain knowledge was also a predominant reason for communication with members not in a task dependency and communication brokers who were also domain knowledgeable served as boundary spanners across application domain areas.

V. DISCUSSION

Strategies to overcome the thin spread of domain knowledge. Laying out a cross-functional communication structure in response to domain knowledge distribution to complement task assignments was a strategy to manage newly formed development relationships with Brazil. The two projects had two different approaches to dealing with the different *distributions of domain knowledge* in their projects, namely through (1) involving new hires in training on the application domain or in reverse engineering its functionality and (2) prescribing cross-functional communication structures that controlled information flow.

SHIP retained application domain knowledge by involving the original US-based developers and 6-month on-site training of new hires. In contrast, APP, which consisted of Brazilian new hires, had the entire development team reverse engineering the applications. Both projects involved development leaders in requirements negotiations with those knowledgeable of the domain, customers.

By designing their cross-functional communication structures, both projects placed team members in key positions to control communication in the project in a manner that was beneficial for disseminating domain knowledge. They were the RAs and dev leaders who were involved in negotiations of requirements with customers; when needed, project members contacted them to clarify requirements or coordinate changes using their knowledge of the application domain. This suggests a strategy of aligning communication to connect domain knowledge experts within the team with those who may lack this knowledge.

Task assignments are not sufficient to explain communication in a project. We observed multiple instances of emergent interactions among team members that fell outside of the assigned tasks. Our findings underscore a necessary trade-off that software projects have to make to balance task assignments and spread of domain knowledge. Because application domain knowledge is spread thinly in organisations, one could easily overload the few people with application domain knowledge by involving them in many tasks [6]. To prevent this overload, some team members are assigned to tasks when they do not have sufficient domain knowledge to complete them.

Despite the differences in these projects, our analysis found that about a third of the actual communication in both SHIP and APP was not related to the task assignments. Our in-depth analysis of these *emergent* interactions suggests that they were either initiated within the mandated cross-functional communication structure by problem solving with those of the same role or with the team leaders; or, by seeking domain knowledge from customers or environment coordinators or those in the same application portfolio, none of which in a task dependency.

However, task assignments should not be ignored, as they appeared to overrule the mandated communication structure. Team members with task dependencies engaged in backchannel communication to complete their tasks. This finding confirms earlier research on informal networks through which employees disregard formal structure in order to increase their productivity (e.g. [28]), and suggests that in these projects the formal structure interfered with the task assignment structure. Identification of such informal patterns is useful in configuring teams and communication structures that optimize access to domain knowledge in projects.

Our findings suggest that socio-technical alignment cannot be sought only from an examination of task dependencies alone, and that there exists a more complex relationship among communication, coordination needs as derived from requirements, the hierarchical control structure and the spread of domain knowledge.

Boundary spanners are communication brokers that have domain knowledge. The three top brokers in the project with multiple applications (APP) also acted as boundary spanners [5] across three of its four application domains. The brokering of communication across domains was facilitated by these project members' roles and position in the projects' cross-functional communication structure. In APP, the top brokers were RAs and the dev leaders who interacted with the customers and acquired information about their business environment beyond one application portfolio. This provided them with a higher level view of the applications and allowed them to communicate across the application domains, similar to other studies' findings of tech leads and architects emerging as boundary spanners because of their broader outlook in the project [39]. Moreover, these brokers' position in the communication structure not only supported the dissemination of domain knowledge within the network but also specifically may have reinforced accumulation of domain knowledge by the broker [40].

VI. THREATS TO VALIDITY

Our evidence of the influence of these organizational structures on the socio-technical alignment is based on self-reported data about communication, and a threat to the internal validity is that the socio-technical interactions were also based on factors such as familiarity with the others or access through physical co-location. Self-reported communication in questionnaires, though a standard practice in research on social networks [35], is open to risks of individual bias and memory and poses a threat to our study's construct validity. We mitigated this threat by using multiple sources of evidence [34]. Besides the survey, we used interviews and direct observations (which provided qualitative accounts of how the teams coordinated) to collect data on the relationships.

Our study also examined only two projects and thus there are limitations to generalizing to other software engineering projects. Our findings apply to projects in which ongoing development of product releases includes forming new teams at remote locations, and where access to application documentation or original designers is limited. Factors that we did not investigate, such as ethnic culture, may have also affected communication in these teams. Research into the effect of cultural factors on requirements communication is a topic for future work.

VII. CONCLUSIONS

Analyzing socio-technical coordination around tasks and their requirements-based dependencies allowed us to study the alignment of communication involving multiple roles with the technical tasks in these two projects. Our exploratory study reveals the importance of examining not only communication that occurs around assigned work, but also communication that emerges from unexpected dependencies, often driven by domain knowledge seeking behavior. We identified that:

- Team members tended to communicate with others who have similar domain knowledge
- Surprisingly, team members followed the project's cross-functional communication structure to gain this knowledge
- Also surprisingly, task dependencies alone were not enough to explain communication flow in a project
- Domain knowledge experts were communication brokers and boundary spanners across multiple applications

Because domain knowledge is fundamental to effective software development, we hope that both researchers and practitioners pay close attention to how domain knowledge experts influence communication patterns in software teams.

Implications for research. As our study was an exploratory study, more research must be done on domain knowledge distribution and cross-functional communication in software projects. Future methods for the analysis and measurement of socio-technical alignment should consider emergent communicators. Emergent interactions may indicate a deficiency of sufficient domain knowledge within a team, and the presence of the experts could be made more visible for improved leverage of their knowledge.

Similarly, the satisfaction of coordination through indirect paths, specifically with brokered communication should also be considered when examining alignment. Coordination can be brokered as a result of adherence to cross-functional communication structures or access to knowledgeable members. This can be integrated in measures of socio-technical congruence by modeling transitive connections. If communication between A to B and B to C is observed, where B is a broker, one can model the transitive connection from A to C as *brokered communication,* and indicate that this communication link satisfies a corresponding *coordination need*. Indirect paths for coordination have not been considered in previous studies of socio-technical alignment and their presence should be studied in relationship to project success.

Implications for practice. Our analysis of socio-technical alignment considered structures other than task assignments. A practitioner can use our findings to improve communication within a software engineering organization. For example, our findings indicate that domain knowledge experts are rare and should be accessible. When organizing teams, managers should give particular attention to potential brokers by identifying members that have exceptional knowledge of the particular application domains or components in the system. Communication structures should strive not only to ensure that these experts are not overwhelmed with emergent interactions, but also to make them accessible for domain knowledge dissemination. When the communication structure intersects with task assignments, team members will engage in backchannel communication to complete their work.

The presence of backchannel communication is not necessarily problematic but may indicate areas where the task assignment was not congruent with either the cross-functional communication structure or the distribution of domain knowledge in the team. Software managers can optimize the team configuration when forming new teams, especially in outsourcing relationships as those we studied.

By understanding the interrelationships between task assignments, cross-functional communication structure, and domain knowledge, organizations gain an increased ability to facilitate communication and coordination in their projects. We hope this will lead to software projects with less wasted communication, less coordination overhead, and increased leverage of the diversity in project roles and domain knowledge.

ACKNOWLEDGMENT

We thank the project members for their time in our research, and NSERC Canada for funding this project. We also thank Gail Murphy, Margaret-Anne Storey, and Jane Cleland-Huang for their comments on early drafts of this paper.

REFERENCES

[1] I. Kwan, A. Schroter, and D. Damian, "Does socio-technical congruence have an effect on software build success? A study of coordination in a software project", IEEE Transactions on Software Engineering, 37(3), May-June 2011, pp. 307-324.

[2] M. Cataldo, P. Wagstrom, J. Herbsleb and K. Carley, "Identification of coordination requirements: implications for the design of collaboration and awareness tools," in CSCW, 2006, pp. 353–362.

[3] D. Damian, L. Izquierdo, J. Singer and I. Kwan, "Awareness in the Wild: Why Communication Breakdowns Occur", In Intl Conf. on Global Software Engineering, 2007, pp. 81-90

[4] M. Cataldo, J. Herbsleb and K. Carley, "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity", in ESEM, 2008, pp. 2–11.

[5] B. Curtis, H. Krasner and N. Iscoe. "A field study of the software design process for large systems". Comm. ACM, 31(11), 1988, pp. 1268-1287.

[6] Reinersten, R. *Managing the design factory*, Free Press, 1997

[7] A. Boden and G. Avram, "Bridging knowledge distribution - The role of knowledge brokers in distributed software development teams", in CHASE, 2009, pp. 8-11.

[8] A. Milewski et al. "Guidelines for Effective Bridging in Global Software Engineering," in Intl Conf. on Global Soft. Engineering, 2008, pp. 23-32.

[9] P. Hinds, and C. McGrath, "Structures that work: social structure, work structure and coordination ease in geographically distributed teams," in CSCW, New York, NY, USA, 2006, pp. 343–352.

[10] Bird, C., D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," In Int. Symp. on Foundations of Soft. Eng., 2008, pp. 24–35.

[11] Cataldo M. and K. Ehlrich, "The impact of communication structure on new product development teams", In ACM CHI, Austin, USA, 2012, pp. 3081-3090.

[12] N. Nagappan, B. Murphy, and B. Basili, "The influence of organizational structure on software quality", Intl Conf. on Soft. Engineering, 2008, pp. 521-530.

[13] J. Herbsleb, Global software engineering: The future of socio-technical coordination, in Future of Soft. Eng., 2007

[14] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in Intl Conf on Soft. Engineering, 1-11, 2009, pp. 1–11.

[15] M. Cataldo and J. Herbsleb, "Factors leading to integration failures in global feature-oriented development: An empirical analysis", in Intl Conf on Soft. Engineering, 2011, pp. 161-170.

[16] K. Crowston, K. and J. Howison, "The Social Structure of Free and Open Source Software Development" First Monday.10(2), 2005.

[17] D. Damian, I. Kwan and S. Marczak, "Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people", in Collaborative Software Engineering, Springer-Verlag, 2010, pp. 57-76

[18] B. Nuseibeh and S. Easterbrook. "Requirements Engineering: A roadmap", in Future of Software Engineering, 35-46, 2000

[19] C. Williams, P. Wagstrom, et al, "Supporting enterprise stakeholders in software projects", in CHASE 2010.

[20] S. Marczak, I. Kwan, and D. Damian, "Investigating collaboration driven by requirements in cross-functional software teams", In Understanding and Softskills, Collaboration and Intercultural Issues Workshop , 2009, pp. 15-22

[21] S. Marczak and D. Damian, "How interaction between roles shapes the communication structure in requirements-driven collaboration", In Intl Conf. on Requirements Eng, 2011, pp. 47-56

[22] K. Ehrlich and K. Chang, "Leveraging expertise in global software teams: Going outside boundaries" In Intl Conf on Global Software Engineering, 2006, pp. 149-158

[23] I. Kwan and D. Damian, "The hidden experts in software-engineering communication", In Intl Conf on Soft. Engineering, 2011, pp. 800–803.

[24] R. W. Helms, "Application of knowledge network analysis to identify knowledge sharing bottlenecks at an engineering firm". In J. Ljungberg amd M. Andersson (Eds.), Intl European Conf. on Inf. Systems, 2006, pp. 1877-1889

[25] R. L. Daft. *Organization theory and design* (10th ed.). Mason, Ohio: South-Western Cengage Learning, 2010.

[26] C. Bird et al., "Putting It All Together: Using Socio-technical Networks to Predict Failures," In Symp. on Software Reliability Engineering, 2009, pp. 109 –119.

[27] D. Krackhardt and R. N. Stern, "Informal networks and organizational crises: An experimental simulation", Social Psychology Quarterly, 51 (2), 1988, pp. 123-140.

[28] R. Cross, A. Parker, L. Prusak and S. Borgatti. Knowing what we know: Supporting knowledge creation and sharing in social networks. Org. Dynamics, 30(2), 2001, pp. 100–120.

[29] D. Berry, "The importance of ignorance in requirements engineering," J. Syst. Softw., 28(2), Feb. 1995, pp. 179–184.

[30] J. Coughlan and R. Macredie, "Effective communication in requirements elicitation: A comparison of methodologies", Requirements Engineering, vol. 7, no. 2, 2002, pp. 47–60.

[31] A. Johri, "Boundary spanning knowledge broker: An emerging role in global engineering firms". Frontiers in Education Conf, Saratoga Spring, 2008.

[32] C. D. Rosso, "Comprehend and analyze knowledge networks to improve software evolution". J. of Software Maint. and Evolution Research and Practice, 21(3), 2009, pp. 189-215

[33] R. S. Burt, "Structural Holes and Good Ideas," American Journal of Sociology, vol. 110(2), 2004, pp. 249-399.

[34] R. K. Yin, *Case study research: Design Methods* (2nd edition), London: Sage Publications, 1994.

[35] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, 1994.

[36] P. Eades and X. Lin, Spring algorithms and symmetry.Theoretical Comp. Science, 240(2), 2000, pp. 379–405

[37] M. Fershtman, "Cohesive group detection in a social network by the segregation matrix index", Social Networks, 19(3), 1997, pp. 193–207.

[38] N. Eagle, A. Pentland and D. Lazer, "Inferring social network structure using mobile phone data". PNAS, 106(36), 2009, pp. 15274–15278

[39] K. Ehrlich et al. "An analysis of congruence gaps and their effect on distributed software development, Intl workshop on Socio-technical congruence, 2008

[40] D. Dekker, F. Stokman, and F. Hans Franses. "Broker Positions in Task-Specific Knowledge Networks: Effect on Perceived Performance and Role Stressors in An Account Management System" ERIM Report Series: ERS-2000-37-MKT, 2000