How Interaction between Roles Shapes the Communication Structure in Requirements-Driven Collaboration

Sabrina Marczak
Software Engineering Global interAction Lab - SEGAL
Department of Computer Science, University of Victoria
Victoria, BC, Canada
smarczak@cs.uvic.ca

Daniela Damian
Software Engineering Global interAction Lab - SEGAL
Department of Computer Science, University of Victoria
Victoria, BC, Canada
danielad@cs.uvic.ca

***Abstract*—Requirements engineering involves collaboration among many project team members. Driven by coordination needs, this collaboration relies on communication and knowledge that members have of their colleagues and related activities. Ineffective coordination with those who work on requirements dependencies may result in project failure. In this paper, we report on a study of roles and communication structures in the collaboration driven by interdependent requirements in a software team. Through on-site observations, interviews with the developers and application of social network analysis, we found that there was significant communication between diverse roles in the project, and identified what were the reasons for communication between the different roles. We also found that these interactions typically involved a core of requirements analysts and testers in close communication, that most often they involved critical members whose absence, whether temporary or permanent, would disrupt the information flow if removed from the project, as well as that new hires were mostly isolated from the team collaboration. Most interestingly we found that the emergent communication structure between the different roles in the project did not conform to the planned communication structure prescribed by the organization. These findings further our knowledge about collaboration driven by requirements, and provide some useful implications for research and development of collaborative tools to support the effective coordination of cross-functional teams in software development.**

***Keywords*-requirements-driven collaboration; coordination; communication structure; communication patterns; interdependent requirements; cross-functional teams; social network analysis; case study.**

I. Introduction

Requirements engineering (RE) drives the software lifecycle from the elicitation phase, to analysis and design, and down to implementation and testing [1]. As such, it involves continuous collaboration among members from different functional groups such as requirements analysis and design, software architecture, development, and testing. Project members communicate to seek domain knowledge or to establish a common understanding about the work to be done. Ongoing coordination is also necessary to manage dependencies with those working on artifacts related to requirements. Implementing interdependent requirements creates a greater need to coordinate across different functional groups in an organization. Project members playing different roles and holding specialized knowledge about different parts of the system need to coordinate effectively to avoid failures. Ineffective coordination with those who work on dependencies may result in failures [1].

Our early studies of collaboration driven by requirements [17] introduced the concept of a requirements-centric social network (RSCN) as the representation of collaboration of a cross-functional team of people whose work is related to particular requirements. We characterized RCSNs in terms of their size and ability to maintain awareness of work, specially when the project team was geographically distributed. We continued with a study of collaboration driven by interdependent requirements [25] in which we found that technical leaders were brokers of incoming and outgoing information flow in the social networks formed around interdependent requirements. This leads us to ask about the interactions between different roles in the project and how they affect their communication.

The purpose of the study reported in this paper is to extend our knowledge of roles in requirements engineering, by examining the interactions of people playing different roles and how these interactions shape the communication structure of collaboration driven by interdependent requirements. Is there a way to improve the communication between the different roles in the project to positively affect coordination driven by requirements? Roles in work teams have been studied in organizational behavior literature [7], but largely overlooked in the RE literature. Defining team structures with clear communication channels among various roles are typical ways to facilitate collaboration and coordination in organizations [26].

Our study was guided by the following research questions:

- Are there patterns of communication in collaboration driven by work on interdependent requirements?
- How do the different roles interact in collaboration driven by work on interdependent requirements?

We report on a field study of software development at a large distributed IT organization, and where we had

the opportunity to study a team that had a well-defined team structure and communication structure among the roles in the project. We identify a number of communication patterns between different roles in the project, and reasons for communication between these roles.

Most interestingly we found that the emergent communication structure between the different roles in the project did not conform to the anticipated, planned communication structure prescribed by the organization. We provide some explanations for the patterns we identify and observations that we make as well as describe some implications for future research and development of collaborative tools to support effective coordination in RE.

II. Related Work

Organizational behavior research has long studied organizational and communication structures in organizations. Collaboration and information sharing have been of particular interest since they are critical elements in effective performance of technical work in organizations. The organizational structure is meant to determine not only the division of labor and modes of operation to achieve a work outcome but also the ways in which information between organizational roles should flow. Traditionally, organizations use hierarchical structures to increase efficiency and control, and guide employees to specialize in a few tasks in different functional units or departments [22]. Since Conway's observation in 1968 [11] that system structure is bound to reflect the structure of social interactions in organizations, we have learned that hierarchical or too rigid organizational structures often create obstacles to the movement of information by blocking flows and by promoting overload of information in certain paths [20]. Organizations can enhance their effectiveness by promoting communication outside the formal, hierarchical boundaries [2]. People often find ways to overcome these obstacles by disregarding the organization structure and establishing informal relationships across functions to accomplish tasks fast. These so called 'informal organization networks' can cut through formal reporting procedures to jump start stalled initiatives and meet extraordinary deadlines [24]. Interactions among people involved in informal relationships represent backchannel communication, also known as 'grapevine'.

Informal structures in organizations have thus been also researched extensively (e.g. [12]). Studies find that backchannel communication fulfills a social function in organizations by helping work groups develop more cohesion [3], and is motivated by a desire to achieve speed in acquiring information and accuracy of the work done [18]. 'Lateral' communication that disregards hierarchical structures is used in organizations to cross functional and departmental boundaries [22]. Longitudinal studies of interactions within informal groups reveal patterns of communication and what has been referred to as network structures (e.g. [23], [9]). Social network analysis has become predominant in studying formal and informal organization networks to support strategic collaboration in large organizations and as mechanisms to assess the health of formal or informal structures following organizational restructuring (e.g. [13]).

In RE, Gotel [21] introduced the concept of 'contribution structures' in order to capture the relevant information about agent participation and the contribution relations that have been defined for requirement artifacts. Our definition of requirements-driven collaboration leverages this notion of 'social infrastructure' underlying RE and intends to provide focused investigations and insights into the roles played by these agents and their interactions. In this paper we further our understanding of communication patterns between roles in requirements-driven collaboration in an organization with well-defined organizational and communication structures.

III. Research Methodology

Our field study used a mixed-methods approach to data collection in a software project at a large distributed IT organization. A three month on-site visit was conducted in which we observed team members working in their native environment; as well, as we constructed communication social networks to identify interdependent requirements-driven collaboration patterns.

A. Case Study Setting

The project we investigated, named *KnowHow*, a fictitious name, is a software maintenance project at a multinational IT manufacturing company. This company develops software to support its business processes. Software is developed and maintained in its development centers located in Brazil and India, as well as in the headquarters office located in the United States. The project's customers are the company's employees, business partners, or contractors.

Business area and project goal. KnowHow is a project that enhances and maintains a group of internal applications used by product management and sales. With over one hundred applications within a group organized into four major groups, the project investigated was the first quarterly release since transfer of the code ownership to the Brazilian center. Because the project was new to the center, the team was under pressure from management to demonstrate ability to deliver on time and with high quality.

RE process. The team maintains a list of desired improvements. These desired improvements address business process changes that the business partners have identified. Each of the four main groups has its own business partner representative, who informs the requirements analysts about the desired improvements. An additional business person manages the business partners. By quarter, requirements analysts select and prioritize the improvements that will form

the quarter release scope. The business manager is the focal point in solving issues and in prioritizing the improvements. After translation into high-level software requirements, the selected improvements are added to the existing product documentation. Each requirement is allocated to one of the hundreds of applications maintained by the *KnowHow* team. The requirements are formally approved and specified within the first three weeks of the release cycle, and serve as the project release contract. Either the business partners or requirements analysts request changes to the requirements, and requirements analysts conduct impact analysis.

The team acquired knowledge about the product and its original requirements through a process of reverse engineering. Formerly conducted by the headquarters office, the *KnowHow* project was transferred to Brazil five months before we started the investigation. Most of the applications had no documentation about requirements, architecture, and their mutual interface. The majority of the employees who had originally developed the applications had already left the company. To minimize the lack of expertise, since every member in the *KnowHow* project was new to the applications, senior management decided to train the team on the application domain and technical matters by asking them to document the applications' requirements and architecture using reverse engineering. The development team spent three months working on this activity.

Team distribution. The *KnowHow* team consists of 44 members, distributed as follows: 38 in Brazil, 5 in the United States, and 1 in India. The 5 business partners are located in the United States and 1 tester is located in India. Located in Brazil are 2 project managers, 4 requirements analysts, 1 test leader, 6 testers, 5 development leaders, and 20 developers. One requirements analyst also acts as the leader of his team, and one of the development leaders is in charge of managing the development team. All members work full time on the project and about 30% of them are contractors or new hires in the company.

Organization structure, roles, and responsibilities. This project had a well-defined team organization and prescribed communication channels between the roles, as shown in Figure 1. This organizational structure demanded that developers and testers do not communicate directly. They have to communicate with their leaders to reach someone in the other functional group, and the leader will discuss directly the matter with the target person.The project managers supervise the overall work and report progress to senior management. The business partners are responsible for identifying the desired improvements and for registering these improvements. The requirements analysts analyze and prioritize the improvements with the help of the business partners and the project managers. The development leaders discuss the improvements when consensus about scope is not reached. Once the scope is agreed, the requirements analysts translate the selected improvements into high-level software requirements, which are formally documented.

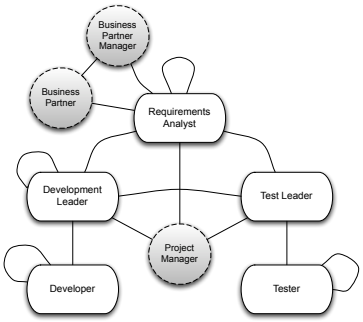

Figure 1. Organizational structure defined for the *KnowHow* project

The testers actively engage in the project after approval of the requirements. The test leader is responsible for designing tools to automate test cases and the testers are in charge of writing them. Developers are responsible for coding the requirements and performing integration tests of their code with other developers' code related to the same requirement.

B. Data Description, Collection and Analysis Methods

The focus of our data collection and analysis was the collaboration around sets of interdependent requirements. We use the term *cross-functional team* to refer to the group of project members who played different roles and worked on artifacts related to the interdependent requirements. These roles included requirements analysts, developers, and testers. We refer to the interactions among different roles in the cross-functional team as *cross-functional interactions*. We use the *RCSN* concept (requirements-centric social network) [17] to represent the collaboration within cross-functional teams associated with sets of interdependent requirements. The nodes in the network represent the members in the cross-functional team and the ties indicate communication interactions among them. A directional tie is drawn if one person reported that communication occurred about a certain requirement of the dependency set with another person.

We spent three months on site collecting data. We used document inspection, interviews, questionnaires and on-site observations to collect our data. We describe each of these methods to explain how we identified the requirements and their dependencies, and their associated requirements-centric networks. To identify communication patterns within these networks and interactions among project roles we used insights from the observations and questionnaires, and methods from social network analysis [27].

Document inspection. To identify the requirements and their dependencies, we inspected requirements documents such

as the requirements specification and the requirements-traceability matrixes. We identified 20 high-level software requirements and 4 sets of requirements dependencies. These dependencies are of constraining nature (e.g., requires, and conflict-with dependencies) as defined by Dahlstedt [15]. An example of 'requires' type of dependency in our study involved the two following fictitious requirements: *Requirement 1* defined that delivery of computer parts requested by customers after acquiring a computer shall be managed by the Part Notice Sales Management system, and *Requirement 2* defined that the system shall send an e-mail to the customer to notify when the part has been shipped from the manufacturing company. The number of requirements per set varied from 2 to 4, with average 2.5 requirements per set. We interviewed the requirements analysts to validate the sets of dependencies identified.

To identify those individuals and the roles assigned to work in every task related to each requirement listed on the sets of dependencies, we inspected project planning documents and built a list of members associated to each requirement. A total of 10 members (out of 45) were listed, as follows: 2 requirements analysts, 1 test leader, 4 testers, 2 development leaders, and 2 developers. On average, each set of dependencies had 4.75 members assigned.

Semi-structured interviews and observations. To develop an in-depth understanding of the project context and to identify communication patterns, we conducted semi-structured interviews and observed the team members working in their native environment. Each of the 10 team members was interviewed at least once individually. Group interviews followed to clarify discrepancies about requirements dependencies or technical information about the project.

Daily for three months, we observed team members performing their activities to identify who collaborated with whom and how they related their tasks and interactions back to requirements. We observed interactions among members in individual situations or in group meetings and notes were taken. Each member was individually shadowed at least three times in distinct phases of the development cycle.

Questionnaire. To refine and corroborate the data about communication interactions collected through interviews and observations, we applied a questionnaire. Questions provided the respondent with a list of names (those assigned to work on artifacts related to the same dependent requirements) and asked the respondent to indicate whom he or she communicated with and the reason for the communication. We provided a list of four reasons, identified as relevant through interviews: requirements negotiation, requirements clarification, communication of changes, and coordination of activities. Appendix A presents the questions.

With this information about communication between the different members in the project we then constructed the RCSNs for each reason for communication. For example, if team member P_1 reported that he or she communicated with P_2 about reason *requirements negotiation* for requirement R_1, then we created a directed tie between P_1 and P_2 in the R_1 *requirements negotiation* RCSN. Out of the 10 participants, 8 responded to the questionnaire. The *available-case analysis* technique [28] was used to replace missing social network data.

IV. Interdependent Requirements-Driven Collaboration Patterns

The analysis of collaboration around the 4 sets of interdependent requirements in this project provided us with 16 requirements-centric networks (4 sets of dependencies times 4 reasons of communication)[1]. In this section we describe and discuss the communication patterns in requirements-driven collaboration identified in this project. By *communication pattern* we mean the recurring repetitions of the same collaboration behavior across the social networks of dependent requirements.

Collaboration driven by interdependent requirements includes significant cross-functional interactions

Although we expected that project members would speak more with colleagues in their own functional teams than with those in other teams, in this project we found the opposite. The functional teams that we observed were the requirements analysis, development, and testing teams. In Figure 2 we plot the amount of within-team vs. cross-team communication in all the networks we studied. One can easily see that the cross-team communication of requirements clarifications, negotiation, and communication of changes far outweighed the within-team communication. Moreover, communication of changes was the most-often reported reason for cross-team communication. As expected, the coordination of activities occurred more often within- than across-teams.

This is an interesting finding when interpreted in the context of the prescribed communication structure in this project shown in Figure 1. To support a project team becoming familiar with a new project, the organization prescribes that the cross-functional communication take place through the development and testing leaders. We observed, however, a predominant direct communication between developers and testers, in what would be termed as backchannel communication in the organizational behaviour literature (e.g., [20]). We observed testers talking with requirements analysts more frequently than talking among themselves. In interviews, team members reported that physical collocation and weekly

[1]Find all networks for this study at http://home.segal.uvic.ca/~smarczak/2011_RE/

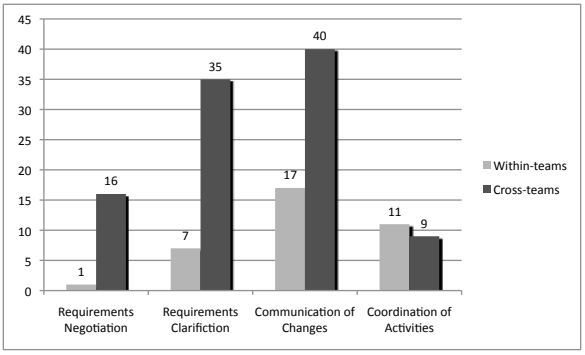

Figure 2. Within- and cross-team communication across all networks

or daily meetings encouraged collaboration with those working on interdependent requirements in different teams.

This predominant cross-functional interaction pattern suggests that the requirements analysts, development, and testing teams were well integrated and collaborated with each other to accomplish the project goals despite the fact that most of the members were new to the project. Because senior management was aware that acquiring knowledge about the legacy applications in such a short time was critical, especially for the newcomers, they constantly encouraged the team leaders to offer support to their new teammates. Team leaders often repeated in their weekly team meetings that no one should feel intimated in asking for help as soon as a problem was detected or as a need for clarification arose.

Requirements clarification and communication of changes are the predominant reasons why team members collaborated with colleagues working on dependent requirements

Although the requirements in this project's scope were intensively discussed with the business partners and only a few changes were implemented, in the interviews team members reported that the main reason for requirements-related discussions were to clarify the requirements, to understand the requested changes in the requirements, and to discuss how to perform the related tasks derived from the requirements. Our questionnaire data confirm this finding. Out of 136 reported instances of communication in the networks we created, 47 were about requirements clarifications, and most of them initiated by testers. This empirical finding corroborates with anecdotal knowledge that to understand the project requirements and to perform their tasks efficiently testers need support from the remaining team members. Approaches such as test-driven development [4], where a tester must clearly understand the project requirements before the test cases are written, can be adopted to impose the comprehension of the requirements before the development cycle starts. This strategy will likely bring the test team closer to the requirements definition phase where an early understanding of the requirements may diminish the need for clarification requests. Studies of teams who followed the test-driven development approach shown that these teams were more productive than those who write test cases after the code has been developed [19]. Similarly, a study of RE process improvement in a distributed team [16] found that, when the requirements analysis activities are conducted in cross-functional teams that involve testers in close interaction with developers and requirements analysts, both the communication among project members and the requirements coverage were improved.

Communication of changes was the other most predominant reason (52 out of 136 instances) for interaction in the networks we studied. Interestingly enough, however, most of our respondents indicated that they did not receive notification of changes in a timely fashion, highlighting the importance of timely communication of changes. Lack of awareness of who is working on interdependent requirements or of changes that affect other requirements may affect the team's coordination ability. For instance, members may prepare to discuss important topics in group meetings based on obsolete information, thus disrupting the work flow. In the absence of details about requirements change-related conversations, we can only speculate that it is highly possible that some of this communication was redundant and the result of change notifications having reached project members too late. Researchers and tool developers should pursue improved methods of communication of changes to increase the effectiveness of notifications of changes to those working on affected requirements.

Actual communication structure in the project is very different from the planned communication structure

The data we collected about actual communication in the project and which we represented in our networks reveal that, overall, a large number of team members exchanged information with each other, instead of a few members controlling the distribution of information as expected based on the team structure. This behaviour characterizes a more loose and decentralized structure in the networks in our study. A test of *network centralization* [29] in our networks yielded an average network centralization index of 0.41, suggesting that the networks were in general characterized by the decentralization of information exchange among team members. The *network centralization* measure is an expression of how tightly organized the network is around its most central nodes. Network centralization increases and approaches a value equal to 1 as one member (or a few members) has connections to many others while the remaining of a team is connected to only a few members. The measure decreases and approaches a value equal to 0 as more members are connected to fewer members, and the distribution of ties is more equal among the members. The decentralization we found in these networks suggest that it is

more likely that the team will be less affected or disturbed by the sudden absence of a colleague who mostly communicates (and presumably holds knowledge) about the project.

An interesting aspect of this finding, however, is that this rather decentralized structure is very different from the planned and intended communication structure. As shown in Figure 1, the senior management in this project has intentionally created a centralized organizational structure focused on the leaders (requirements analyst leader, the 5 development leaders, and the test leader) to control information flow and awareness of decisions. An explanation for the fact that the members coordinated directly, through backchannel communication [20], and in a more decentralized manner than expected, relates to the nature of this project. In this maintenance project, knowledge about the applications was built through a reverse engineering activity, in which developers spent time inspecting the applications and documenting their actual requirements and architecture. In addition, requirements analysts had to master the applications through the study of recently written documentation and discussions with business representatives. Therefore, we can explain the communication links being distributed among the diverse roles by the flatter distribution of knowledge about requirements and absence of reliance on key team leaders.

Although the presence of backchannel communication has long been reported in organizations, with both positive and negative connotations [20], the identification of roles that engage in backchannel communication may be useful in making decisions about the effectiveness of prescribed communication structures in projects. A requirements-centric network that exhibits high backchannel communication may also be indicative of requirements interdependencies that are poorly understood or highly volatile.

Core subgroups of members originally assigned to work on the requirements closely collaborate with each other

Although we found that, overall, the networks had a rather decentralized structure, these networks had subsets of project members more connected with each other than with others. These were core subgroups formed primarily of requirements analysts and testers, who closely collaborated with one another. Interestingly enough, these members were among those initially assigned to work on these requirements. This suggests that the initial project plan was useful in this project, a finding that is different from other studies that found that project plans quickly become obsolete [14].

The existence of cores and their membership was corroborated by applying the *core-periphery* [8] test. This test indicates the extent to which the structure of a network consists of two classes of members: the core, in which members are connected to each other in some maximal sense; and the periphery, a class of members that are more loosely connected to the core. The higher and closer to 1 the index, the closer the network approaches a core-periphery structure. The core-periphery index of the networks varied from 0.67 to 1, except three networks which had indexes smaller than 0.35, and were often formed by requirements analysts and testers. For example, Figure 3 shows the core of the *Communication of changes* RCSN for the dependency set $D2$. This network core is formed by the following members, highlighted with a red circle in the Figure: *Requirements Analyst 1*, *Developer*, *Tester 1*, and *Tester 2*. The requirements analyst and tester roles were not directly involved in the reverse engineering activities phase, which may explain why they collaborated so closely. They had to somehow acquire knowledge about the requirements to perform their tasks. Therefore, these members closely exchanged information about the project and collaborated with each other.

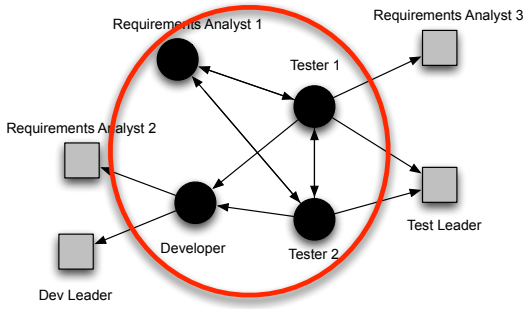

Figure 3. Example of a core within a RCSN

The cross-functional requirements-centric teams contain informal cliques that are also cross-functional

By applying the social network measure *clique* we identified that, whenever cliques were present, they consisted of members playing diverse roles in the project. In social network analysis, a clique [29] is a subset of at least three members of a network in which every possible pair of members is directly connected by a tie and this clique is not contained in any other clique. Although we found no cliques in the Requirements negotiation and Coordination of activities networks, there were 2 cliques in each Requirements clarification (8 in total) and 3 cliques in each Communication of changes networks (12 in total).

The cliques identified within the Requirements clarification networks typically consisted of a requirements analyst, a developer, and a tester. For instance, Figure 4 shows the RCSN for dependency set $D2$. Of the two cliques in this network, the first was composed of *Requirements Analyst 1*, *Developer 1*, and *Tester 1*; and the second of *Developer 1*, *Tester 1*, and *Tester 2*. These cliques reflect what we observed while visiting the team on site: testers would walk to the developer desk to ask for clarification, and they would walk together to the requirements analyst office to

discuss the requirements. Note that the requirements analyst involved in this clique is someone who had not been assigned to work on the dependent requirements represented in this network, implying that somehow these members were aware that *Requirements Analyst 1* could help. Since developers spent three months working on reverse engineering to document the applications, it was natural for the unfamiliar testers to seek the developers' help regarding the requirements. The developers acted as the first source of requirements information for the testers.

On the other hand, the Communication of changes cliques were formed by a developer and two testers. One would ask why a developer's involvement in a clique about notification of changes? We would expect that requirements analysts were involved in these cliques because they were very active in negotiating requirements changes and communicating them to the project managers and developers. However, the organization structure prescribed that the development leaders would notify developers of changes to their teammates. It is likely that these notifications took place in group meetings and that the developers then forwarded this information in person to the testers. In fact we had the chance to observe repeatedly, when changes happened, this flow of information exchange about changes to requirements.

The absence of members initially assigned to work on interdependent requirements is most likely to disrupt collaboration about these requirements

In our interviews and observation sessions we noticed that some developers and testers were essential to the project development. They were critical not only for the knowledge they had acquired about the requirements, but also for their ability to point out quickly who was currently working on something related to one's work, where to find certain information, and to whom to go in order to solve a problem. These members were confirmed by the *cutpoint* [29] measure as critical in keeping information flowing (refer to Table I). A cutpoint is an actor (or a set of actors) in a network that, if absent for some reason, would cause the network to be divided into unconnected parts. For example, in Figure 5 the actor labelled as *Dev Leader 2* is a cutpoint. If absent, the network will be then divided in 3 otherwise unconnected groups, which are: *Requirements Analyst 1* and *Requirements Analyst 3*; *Developer 1*, *Dev Leader 1*, *Developer 2*, and *Developer 3*; and *Tester 1* and *Tester 2*.

Table I
MEMBERS WHOSE ABSENCE MAY DISRUPT INFORMATION FLOW

Dep	RN	RC	CC	CA
D_1	Dev	Dev Tester	Dev Tester	Tester Test Lead
D_2	Dev	Dev Tester	Dev Tester	Tester Test Lead
D_3	Dev	Dev Tester	Dev Tester	Tester Test Lead
D_4	Req An Dev	Req An	– none –	Dev Lead

Each cutpoint is a member initially assigned to work on these requirements and Table I shows the role distribution. Similar to brokers in collaboration driven by interdependent requirements [25] – *Dev Leader 1* is also a broker between *Developer 1* and *Developer 2* –, cutpoints mediate communication between other network actors. The absence of a cutpoint, however, appear to have a greater impact on the network, since the absence will disrupt communication between entire groups rather than individuals.

This finding brings implications for human resource management in a maintenance team, where knowledge about the legacy application is a precious asset. A cutpoint member is critical because if he leaves the company or is allocated to work on another project, the network will become disconnected and information will not flow between its disconnected parts. Although it is an implicit characteristic of a working team to find alternative ways to communicate

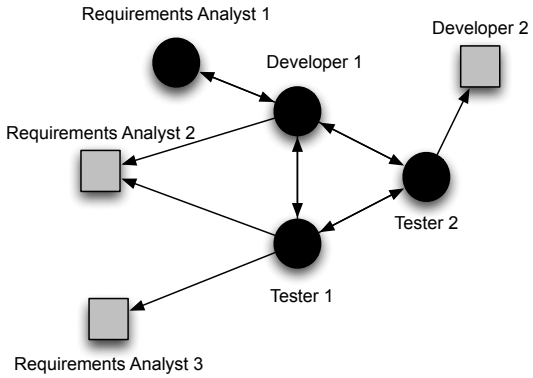

Figure 4. Example of a clique within a RCSN

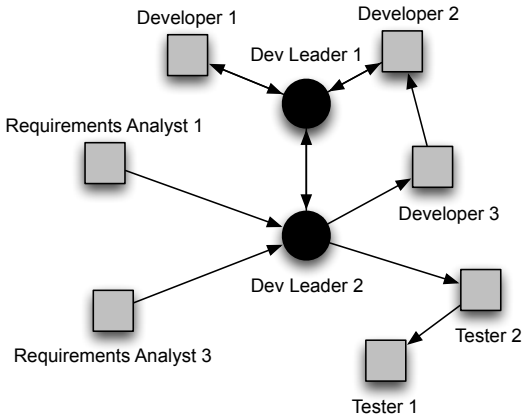

Figure 5. Example of a cutpoint and of a broker within a RCSN

in the absence of a person to mediate, a cutpoint's absence would likely cause disruption of the information flow until the team could mend the broken flow. Both the presence of the cutpoints and the awareness of who they are should be of any manager's interest in order to minimize the consequences of dependence on significant members.

New hires are isolated from the rest in the requirements-driven collaboration

During our on-site visit, we noticed that some members who were assigned to work on interdependent requirements were often isolated from others collaborating on the respective dependent requirements. These *isolated members* were new hires to the company and new to the project and, most often, developers or testers. Their isolation was confirmed when we graphically plotted the networks. As an example, the RCSN in Figure 6 contains an isolated developer, the actor labelled as *Developer 2*. This network represents the communication of changes-RCSN for dependency set D_1. Thus, the isolation of *Developer 2* suggests that he may not have been notified that changes took place for the requirements on which he was working.

Across the 16 networks, we found 17 instances of isolated members, specifically 5 distinct people playing 4 distinct roles, as follows: 9 occurrences of isolated developers (1 single person), 6 occurrences of testers (2 persons), 1 occurrence of requirements analyst, and 1 occurrence of test leader. A single developer was the member most-often isolated across the networks. Although this suggests that this developer did not collaborate with his colleagues, during our on-site visit we attended meetings where he was quietly sitting in the room.

This finding corroborates the evidence that newcomers into a software team have difficulties in knowing how and when to ask questions of others, resulting in communication problems [5]. As people develop a relationship and trust each other, it is common for them to contact colleagues and ask for help or clarification. It is common for members who know each other to exchange information during informal meetings, such as a talk in the hallway or in the cafeteria. When new members do not know who has expertise on each area, or what team members are doing in the project, they may not know who to go to for help or clarification. These people may feel disconnected from the team or even be left out of discussions due to lack of attention of more senior members. Awareness of the isolation of newcomers may flag managers to develop strategies to insert these recently hired members in a faster and more effective way in a team.

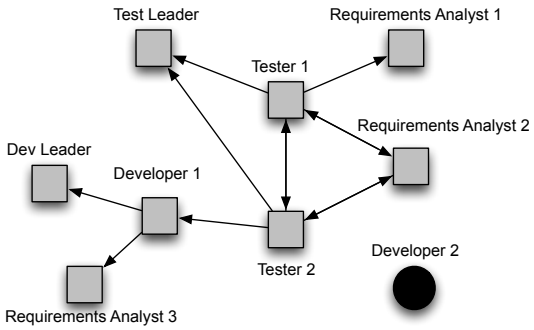

Figure 6. Example of an isolated member within a RCSN

V. Threats to Validity

Internal validity. The communication patterns in our study were analyzed from observations and self-reported questionnaire data. Questionnaires are based on the memory and perceptions of the participants [10]. To minimize the risk of collecting incomplete or unreliable data, we deployed the questionnaire in the beginning of the Testing phase when all team members were still actively working on the project. We believe that our on-site observations over the three-month provided us with sufficiently rich data to cross-validate the questionnaire data and thus allow us to construct networks of reasonable quality.

External validity. Although the report of a single case study can constrain the generalizability of the empirical insights, we believe that the reported patterns may be representative of other projects with similar characteristics. These include the (1) roles played by the project members, (2) the presence of prescribed communication channels between the different functional groups in the organization, (3) the maintenance of legacy applications, and (4) the team's lack of familiarity with the product.

Another limitation is the small number of requirements dependency sets investigated. We could not anticipate the number of dependency sets prior to data collection; otherwise, we would have aimed for a project with a higher number of sets. A larger number of requirements by set may also change patterns of collaboration driven by interdependent requirements. An investigation of larger dependency sets could reveal whether the patterns found would hold.

VI. Final Considerations

In this paper we have described patterns of collaboration driven by interdependent requirements identified in an industrial case study of a maintenance project. These patterns involve interaction among different roles in the project. The patterns presented along with the detailed description of the roles involved in them answer our research questions. To study collaboration among those who need to coordinate due to interdependencies in requirements, we used a mixed-methods approach to data collection and analysis. We also used concepts and measures from social network theory to

triangulate the results obtained from interviews and observations, increasing the reliability of our findings.

In this project we found that there was significant communication between diverse roles in the project, and identified that the most frequent reason for inter-role communication was requirements clarification and communication of changes. We also found that cross-functional interactions typically involved a core of requirements analysts and testers in close communication, that most often they involved critical members whose absence, whether temporary or permanent, would disrupt the information flow in project, as well as how new hires were mostly isolated from the team collaboration. Most interestingly we found that the emergent communication structure between the different roles in the project did not conform to the anticipated, planned communication structure prescribed by the organization. Despite the fact that senior management intentionally imposed an organizational structure where members would have to go to their leaders to communicate with colleagues in other teams, cross-functional communication was predominant– suggesting that members go beyond organizational boundaries to acquire information necessary to perform their interdependent requirements-related tasks.

This study has a number of implications for future research and development of collaborative tools to support effective coordination in RE. Aside from the research implications we discussed in relation to particular patterns identified in our findings, we include below some directions worth pursuing in future research and tool development.

For researchers, there is value in investigating more complex coordination situations. We investigated the coordination in the development of sets of 2-4 interdependent requirements in a maintenance project with a new team. Future studies should investigate larger projects that contain multiple dependent requirements, to obtain insight into the nature of requirements-driven collaboration over complex technical dependencies. Similarly, studies of new development projects should bring insights into requirements-driven collaboration patterns when the requirements and the application are new to the team. In addition, non-functional requirements should also be examined to broaden the current knowledge about requirements-driven collaboration. Non-functional requirements may not have clear boundaries; thus, it may be not as straightforward to identify who is assigned to work on them, potentially making necessary the extension of our methods of identifying RCSNs.

More complex coordination situations are expected when more roles are involved in the management of requirements throughout the project life cycle. In this study, we examined interactions between requirements analysts, developers, and testers, but properties of inter-role communication should be studied further in projects that include other roles such as business analysts, project managers, or software architects.

For tool developers, great potential exists to integrate support for the collaboration and management of cross-functional teams into existing requirements management tools. Tools that automatically generate RCSNs periodically or at certain points in the project could assist both project members and managers in identifying communication patterns similar to those found in this study and in making decisions to adjust RE processes or communication structures if necessary. Tools to generate RCSNs as described in this study automatically could use data-mining techniques [6] [30] as well as automated requirement-traceability tools [28] to identify who works on which artefacts, and to trace these artefacts to requirements. Project and artifact data may be extracted from issue-tracking repositories, requirement repositories, mailing lists, and chat logs, for example.

Our study of roles and communication structures in collaboration driven by interdependent requirements should be complemented by future studies that also benefit from performance criteria in the project and investigate the relationship between patterns in communication between the different roles and the project success. With such information one can design improved RE processes, collaborative tool support, or communication infrastructure in organizations to enable effective coordination in software projects.

APPENDIX

Questionnaire Questions Figure 7 shows a generalized version of the customized communication questions used in the questionnaire.

2 - Communication

Instructions: *For questions 11 to 17, please do not hesitate in acknowledging you have not communicated with someone.*

11. Please identify the names of project team members that you communicate in connection with this project and provide details of interaction. For each name included in "Name of project team member" column, indicate if you communicate or not with this person marking "Yes" or "No" in "I communicated with this person" column. Please fill the remaining columns only for those people you answered "Yes".

Name of project team member	I communicated with this person	Frequency per week *(Indicate how frequently you contact this person)*	Nature of interaction	Requirements *(Indicate all requirements that apply: R1 and R2 as in **Appendix** provided)*
Team member 1	☐ Yes ☐ No	☐ Less than once ☐ Once or twice ☐ 3-4 times ☐ 5 or more times	☐ Communication of changes ☐ Coordination of activities ☐ Requirement clarification ☐ Requirement negotiation ☐ Other:	☐ R1 ☐ R2
Team member 2	☐ Yes ☐ No	☐ Less than once ☐ Once or twice ☐ 3-4 times ☐ 5 or more times	☐ Communication of changes ☐ Coordination of activities ☐ Requirement clarification ☐ Requirement negotiation ☐ Other:	☐ R1 ☐ R2
Team member 3	☐ Yes ☐ No	☐ Less than once ☐ Once or twice ☐ 3-4 times ☐ 5 or more times	☐ Communication of changes ☐ Coordination of activities ☐ Requirement clarification ☐ Requirement negotiation ☐ Other:	☐ R1 ☐ R2

12. Please indicate the names of other additional people that you communicate in connection with this project and provide details of interaction. The people in this list can be allocated in different project teams than yours. The names you provide should not have appeared in Question 11.

Name	Role	Frequency per week	Nature of interaction	Requirements *(Indicate all requirements that apply: R1 and R2 as in **Appendix** provided)*
		☐ Less than once ☐ Once or twice ☐ 3-4 times ☐ 5 or more times	☐ Communication of changes ☐ Coordination of activities ☐ Requirement clarification ☐ Requirement negotiation ☐ Other:	☐ R1 ☐ R2 ☐ Others:
		☐ Less than once ☐ Once or twice ☐ 3-4 times ☐ 5 or more times	☐ Communication of changes ☐ Coordination of activities ☐ Requirement clarification ☐ Requirement negotiation ☐ Other:	☐ R1 ☐ R2 ☐ Others:

Figure 7. Generalized version of the customized questionnaire questions

ACKNOWLEDGMENT

We would like to thank the members of the *KnowHow* project for finding time to contribute to our research. This research is sponsored by NSERC of Canada and a University of Victoria Fellowship. Sabrina Marczak would like to thank PUCRS University, Brazil, where she is currently working, for the time granted for her to conclude this work.

REFERENCES

[1] A. Al-Rawas and S. Easterbrook. Communication Problems in Requirements Engineering: A Field Study. In *Proc. of the Conf. on Professional Awareness in Software Engineering*, pages 46–60, London, England, 1996. Royal Society.

[2] T. Allen. *Managing the Flow of Technology*. MIT Press, Crambridge, United States, 1977.

[3] R. Baron and J. Greenberg. *Behavior in Organizations*. Allyn and Bacon, Boston, United States, 1990.

[4] K. Beck. *Test-Driven Development: By Example*. Addison-Wesley, Boston, United States, November 2002.

[5] A. Begel and B. Simon. Struggles on new college graduates in their first software development job. In *Proc. of the SIGCSE Technical Symposium on Computer Science Education*, pages 226–230, Portland, United States, March 2008.

[6] N. Bettenburg, S. Just, A. Schroter, C. Weiss, R. Premraj, and T. Zimmermann. What Makes a Good Bug Report? In *Proc. of the Int'l Symposium on Foundations of Software Engineering*, pages 308–318, Atlanta, USA, 2008. ACM.

[7] B. Biddle. Recent Development in Role Theory. *Annual Review of Sociology*, 12(1):67–92, 1986.

[8] S. Borgatti and M. Everett. Models of Core/Periphery Structures. *Social Networks*, 21(4):375–395, October 1999.

[9] M. Burkhardt and D. Brass. Changing Patterns or Patterns of Change: The Effects of a Change in Technology on Social Network Structure and Power. *Administrative Science Quarterly*, 35(1):104–127, March 1990.

[10] D. Conrath, C. Higgins, and R. McClean. A Comparison of the Reliability of Questionnaire Versus Diary Data. *Social Networks*, 5(3):315–322, September 1983.

[11] M. Conway. How Do Committees Invent? *Datamation*, 14(4):28–31, April 1968.

[12] S. Crampton, J. Hodge, and J. Mishra. The Informal Communication Network: Factors Influencing Grapevine Activity. *Public Personnel Management*, 27(4):569–584, 1998.

[13] R. Cross, S. Borgatti, and A. Parker. Making Invisible Work Visible: Using Social Network Analysis to Support Strategic Collaboration. *California Mgmt. Review*, 44(2):25–46, 2002.

[14] B. Curtis, H. Krasner, and N. Iscoe. A Field Study of the Software Design Process for Large Systems. *Communications of the ACM*, 31(11):1268–1287, November 1988.

[15] A. Dahlstedt and A. Persson. *Engineering and Managing Software Requirements*, chapter Requirements Interdependencies: State of the Art and Future Challenges, pages 95–116. Number 5. Springer-Verlag, Germany, 2005.

[16] D. Damian and J. Chisan. An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. *IEEE Transactions on Software Engineering*, 32(7):433–453, July 2006.

[17] D. Damian, S. Marczak, and I. Kwan. Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks. In *Proceedings of the International Requirements Engineering Conference*, pages 59–68, New Delhi, India, October 2007. IEEE Computer Society.

[18] K. Davis. Where Did That Rumor Come From? *Fortune*, page 34, 1979.

[19] H. Erdogmus, M. Morisio, and M. Torchiano. On the Effectiveness of the Test-First Approach to Programming. *IEEE Trans. on Software Engineering*, 31(3):226–237, 2005.

[20] D. Fisher. *Communication in Organizations*. West Publishing Company, Minneapolis, United States, January 1993.

[21] O. C. Z. Gotel. Contribution Structures. In *Proc. of the IEEE Int'l Symposium on Requirements Engineering*, pages 100–107, York, England, March 1995. IEEE Computer Society.

[22] P. Hinds and S. Kiesler. Communication Across Boundaries: Work, Structure, and Use of Communication Technologies in a Large Organization. *Org. Science*, 6(4):373–393, 1995.

[23] D. Krackhardt. *Organizations and Networks: Structure, Form, and Action*, chapter The Strength of Strong Ties: The Importance of Philos in Organizations, pages 216–239. Number 8. Harvard Business School Press, Boston, United States, 1992.

[24] D. Krackhardt and J. Hanson. Informal Networks: The Company Behind the Chart. *Harvard Business Review*, (Reprint no. 93406), July 1993.

[25] S. Marczak, D. Damian, U. Stege, and A. Schroter. Information Brokers in Requirement-Dependency Social Networks. In *Proceedings of the International Requirements Engineering Conference*, pages 53–62, Barcelona, Spain, September 2008. IEEE Computer Society.

[26] D. Pugh, D. Hickson, and C. Hinings. An Empirical Taxonomy of Structures of Work Organizations. *Administrative Science Quarterly*, 14(1):115–126, March 1969.

[27] J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, London, England, 2nd edition, March 2000.

[28] D. Stork and W. Richards. Nonrespondents in Communication Network Studies: Problems and Possibilities. *Group and Organization Management*, 17(2):193–209, 1992.

[29] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Crambidge University Press, Crambidge, United Kingdom, 1994.

[30] T. Wolf, A. Schroter, D. Damian, L. Panjer, and T. Nguyen. Mining Task-Based Social Networks to Explore Collaboration in Software Teams. *IEEE Software*, 26(1):58–66, Jan. 2009.