On the Understanding of Requirements-Driven Collaboration:
A Framework and an Empirical Field Investigation

by

Sabrina Marczak
B.Sc., Pontifícia Universidade Católica do Rio Grande do Sul, 2001
M.Sc., Pontifícia Universidade Católica do Rio Grande do Sul, 2003

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

On the Understanding of Requirements-Driven Collaboration:
A Framework and an Empirical Field Investigation

by

Sabrina Marczak
B.Sc., Pontifícia Universidade Católica do Rio Grande do Sul, 2001
M.Sc., Pontifícia Universidade Católica do Rio Grande do Sul, 2003

Supervisory Committee

Dr. Damian, Supervisor
(Department of Computer Science)

Dr. Coady, Departmental Member
(Department of Computer Science)

Dr. Gaines, Departmental Member
(Department of Computer Science)

Dr. Stege, Departmental Member
(Department of Computer Science)

Dr. Helms, Outside Member
(Department of Information and Computing Science, Utrecht University)

Supervisory Committee

Dr. Damian, Supervisor
(Department of Computer Science)

Dr. Coady, Departmental Member
(Department of Computer Science)

Dr. Gaines, Departmental Member
(Department of Computer Science)

Dr. Stege, Departmental Member
(Department of Computer Science)

Dr. Helms, Outside Member
(Department of Information and Computing Science, Utrecht University)

ABSTRACT

Requirements engineering is at the heart of software engineering, and as such, it involves collaboration among many project team members. This collaboration is driven by coordination needs and 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. To study the coordination of those who need to coordinate work due to interdependencies in requirements, this dissertation introduces the concept of requirements-driven collaboration as collaboration that occurs during the elicitation, definition, specification, implementation, testing, and management of requirements. The first contribution of this research is a framework to study requirements-driven collaboration. This framework is based on

social network theory and provides techniques to study communication and fleeting knowledge that underlying collaboration driven by requirements. This framework was incrementally developed throughout the research, first informed by literature review and then empirically-informed through its application in case studies of requirements-driven collaboration. The initial, literature-informed version of the framework was used to guide the design of empirical investigation of requirements-driven collaboration in two globally-distributed software development projects. The framework was then revised based on the insights obtained from its application in the two projects. The empirical evidence about requirements-driven collaboration in these projects represent the second major contribution of this dissertation. Among others, I identified that for both projects the membership of the requirements-driven social networks are dynamic and include important cross-site and cross-team interactions, that the power of distributing knowledge is not in the hands of few team members, and that there are members brokering information between two otherwise unconnected colleagues. The research in this dissertation brings implications for requirements engineering and for the study of collaboration is software projects. By providing researchers and practitioners with a set of techniques to study and evidence about communication and fleeting knowledge in requirements-driven collaboration, the framework offers a mechanism to examine fine-grained details in software projects. The insights obtained can be used to reason about how tools and processes can be improved to better support collaboration throughout the development life-cycle.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

Before I thank those who guided me through, helped me with, and participated in my PhD journey, first I would like to thank those who were my role models and the reason I succeeded in getting here in the first place.

Dad, for your example of courage and of dignity, for teaching me how to respect and to love myself, for showing me how to fight for what I want and believe, for dreaming with me, and for eternally and tirelessly "shaping" me.

Lucia Giraffa, my dear Master supervisor and for life friend. You are the reason I chose to pursue academic life. Your passion for the things you do, your belief that education can change one's life for better (my experience confirms it is worth believing it), your determination and exemplary actions, your patience and supportive words made me want to be like you. You will be my eternal mentor in life.

Jorge Audy, my "guru". You introduced me to the Software Engineering world in the best way one could wish it happens. You gave me the chance to experience theory and practice together, and from this experience I learned that they supplement and need each other. A researcher needs to be aware of this dependency. You guided me through my first steps in industry. You trusted I could do things when I had no clue on how to start working on them. The visionary way you see things, and the simple way you deal with and solve problems has been inspiring me to constantly challenge myself aiming to learn more, to experience more, and to try to be a better professional. I am glad you found me. It amazingly changed my life.

Now, I would like to thank those who were directly involved in my PhD journey.

My supervisor, professor Daniela Damian, also kindly known as *Dana*. Over these last years you were not only my research supervisor but also a great friend. I am forever thankful for your support in getting me engaged in a new society and culture. You smoothed the adaptation process on a way that only someone who experience herself could have done it. Through the years I was lucky to learn from you that it is possible to be a wife and a mom and still a successful professional. Work-life balance is not something easy to achieve but

it is amazing when one benefits from it. You are the demonstration that it is worth pursuing this balance. Thank you for your patience in repeating over and over the same advices until I matured enough to understand them. Thank you for the hard words. Despite the fact that sometimes they hurt, for sure they only did good. Thank you for caring about me like a mom care for her child.

The dear supervisory committee members, professor *Morgan Baker (in memoriam)*, *Yvonne Coady*, *Brian Gaines*, and *Ulrike Stege*, a special thank you for helping me to see what was in front of my eyes and I could not find words to define. Your knowledge in diverse fields made this an interesting interdisciplinary experience for me. It was great to learn and to receive advice from you. Thank you for the time and attention you gave me when, with no doubt, you had work with higher priority to accomplish. Professor *Remko Helmes*, thank you for accepting the challenge of coming on board to replace Dr. Baker and having to come to speed with the work in such a short time. Professor *Helen Sharp*, thank you for the interest on my work.

My dear family, *dad*, *mom*, *brother*, *Marga*, and *Lennon*, for your love and care even at a distance. The hole in my soul from being apart from you was diminished with your smiles and funny stories every time we saw each other online or talked on the phone. *Magno, Mirna, and Ericka*, thank you for becoming my family in Canada. You will be forever in my heart.

Friends and colleagues, *Rafael Prikladnicki* and *Tiago Ferreto*, for sharing the joy and lessons learned throughout your journeys to become independent researchers; *Thanh Nguyen-Huynh*, *Irwin Kwan*, and *Adrian Schröter*, for the unforgettable good memories as students (I am looking forward now for building more memories as research collaborators); *Maryam Daneshi*, *Lars Grammel*, *Maria-Elena Hernandez*, *Matthew Richards*, *Christoph Treude*, *Jennifer Wong*, *Yanyan Zhuang*, for showing me a bit of your culture and expanding my horizons about life.

University of Victoria and Computer Science staffs, life was made easier with the help received from you from the registration process to the defence scheduling time. The University of Victoria care with international student is outstanding and deserves recognition from those who are helped by the always smiling

staff members. The International Affairs Department, the Teaching and Learning Centre, and the Writing Centre in special offered substantial support during my first months in Canada. Special thanks to the Computer Science secretaries *Wendy Beggs*, *Isabel Campos*, *Nancy Chang*, and *Carol Harkness*. So many times you clarified guidelines for me, helped me to find how and where to go to solve issues, and patiently guided me to find my way on campus until I got used to it. Also, thank you for your help with the Computer Science Volunteer Program. Teaching Internet for seniors was challenging but also an amazing experience. The volunteer group could not have done it without your volunteer support.

My dearest "care-giver friends", a very special thank to you for helping me survive an unexpected health problem. They are: *Dana*, the expert on the health problem; *Dad*, the one who helped me to keep my head up; *Marga*, my angel on Earth who did not let me give up; *Buque*, the special messenger; *Isabel*, the everyday "how are you doing? do you need something?" neighbor; *Mirna and Magno*, for helping like only parents would help; *Bill*, the wonderful great cook and Irish neighbor; *Patty*, the example that one can make it, does not matter the challenges that life puts in front of you; *Maryam*, the cheer-up friend and weekly walking company; *Adrian and Irwin*, the kindest and most helpful colleagues ever; and *Marietta and Steve*, for opening your house to me and being so kind. Last but not least, *Matt* for your patience and support.

NSERC and University of Victoria Graduate Fellowship Program, for funding my research. Because of your funding I could concentrate full-time on my studies and become what I am today.

God, for blessing me and for sending His angels to watch me.

The ultimate measure of a man is not where he stands in moments of comfort and convenience, but where he stands in times of challenge and controversy.
Martin Luther King Jr

DEDICATION

To those who taught me that the end of the road is not all that matters. The way you got there, what you learned on the way, and how you changed those you met on the way are one of the most valuable things in one's life. I am absolute sure that you know who you are!

Chapter 1

Introduction

Requirements engineering plays an important role in a project life-cycle, since requirements drive the development of the subsequent project phases. The later in the development life-cycle that a software error is detected, the more expensive it is to repair it [41]. Therefore, it is cost effective to define and specify requirements early on in the project, and to manage them throughout the development cycle.

In order to develop the requirements, cross-functional software teams composed of members representing different functional groups need to establish a shared understanding or a common ground about the requirements. Effective communication and fleeting knowledge (also known as awareness [52]) is important to foster this common ground [31]. Moreover, communication and fleeting knowledge are important for the coordination of work necessary to develop the project requirements since requirements engineering involves highly collaborative activities. For example, requirements analysts collaborate intensively with business partners to gather, to specify, and to negotiate requirements.

Coordination in general is the act of managing interdependencies between activities [95]. In order to coordinate properly, team members need to maintain up-to-date knowledge about the requirements, to exchange information about tasks related to a certain requirement, and to propagate information about changes on requirements. In addition, they need to be able to locate expertise when help is necessary to complete their tasks [48] [54], and to identify who is currently available to help [52].

Although benefits of coordination in software teams were investigated in previous research, there is a lack of understanding about communication and fleeting knowledge in the coordination necessary in the development and management of software requirements. This dissertation aims to further the understanding of this collabora-

tion, which I name *requirements-driven collaboration* from here onwards.

1.1 Research Approach

In this section I present the research problem that this dissertation addresses, the research goal and the derived research questions, the investigative approach defined to address the dissertation goal, and the two industrial projects that I empirically investigated to answer the research questions.

1.1.1 Problem Statement

To effectively communicate and have up-to-date fleeting knowledge is challenging in requirement engineering because members of cross-functional teams have different backgrounds and understanding about requirements. The geographical and organizational distances are factors that contribute to increased challenges faced by the team members. The coordination of work related to a requirement or to interdependent requirements is impacted when these members do not have a shared understanding or common ground about the set of requirements they are working on. Researchers and practitioners lack knowledge about how cross-functional software teams composed of multiple and diverse roles collaborate to develop work driven by requirements. More specifically, little is known about how members of these teams communicate and maintain fleeting knowledge in the coordination necessary to develop software requirements. A better understanding of requirements-driven collaboration is needed to enable reasoning about it so then better tools, processes, and practices to support collaboration throughout the development life-cycle can be developed.

1.1.2 Research Goal and Questions

Research goal: The goal of this research is two-fold. First, my goal is to develop an approach to study requirements-driven collaboration and specifically communication and fleeting knowledge. Second, I aim to further the understanding of requirements-driven collaboration by empirically examining communication and fleeting knowledge in the coordination necessary to the development of requirements in industrial projects.

Research questions: The posed research questions to address the two-fold research goal are presented below and discussed in details next.

1. How can one investigate requirements-driven collaboration?
2. What are typical characteristics of communication and fleeting knowledge networks established during collaboration driven by requirements?
3. What structures do communication and fleeting knowledge networks have in requirements-driven collaboration?
4. How do team members exchange information and share knowledge relevant to performing work driven by requirements?
5. To what extent are requirements-driven collaboration patterns aligned with coordination needs informed by requirements dependencies? Moreover, to what extent do requirements-driven collaboration patterns follow the organizational structure?

Research question 1: *How can one investigate requirements-driven collaboration?*

Requirements engineering is important since it establishes what needs to be designed and defines a baseline for the following phases of the project life-cycle. The requirements are then used by the software team to develop, to test, and to deploy the software system in the user environment. As such, those involved in the many activities that relate to requirements play different roles throughout different phases of the development life-cycle, and belong to different organizational functions. These members also need to communicate and to share a common understanding about the requirements in order to successfully develop the product and to attend the stakeholders needs. Research in software engineering has traditionally focused in investigating collaboration that takes place among developers, leaving out other important roles that contribute and are essential for the project development, such as requirements analysts and testers. In addition, the literature mainly reports on which mechanisms software teams adopt to coordinate work, what causes coordination problems in software projects, how communication breakdowns impact collaboration necessary to team members perform work, and how coordination and collaboration issues affect software productivity and quality. To the best of my knowledge, there is no research that describes how team members coordinate work necessary to the

development and management of requirements emphasizing the traceability from requirements to downstream software artifacts. The lack of such research implies the absence of a method to investigate collaboration among members of cross-functional software team at a fine-grained level. This question aims to define an approach for studying requirements-driven collaboration in such desirable level.

Research question 2: *What are typical characteristics of communication and fleeting knowledge networks established during collaboration driven by requirements?*

Requirements are usually volatile [36] [18], thus changes to requirements need to be communicated promptly to team members to avoid negative impacts on quality and team productivity. Cross-functional software teams, especially large and distributed ones, often have difficulty coordinating requirements development and identifying team members that are knowledgeable of certain features. The lack of awareness of who is working on related activities or of who can help may affect the members ability to coordinate work. This question aims at characterizing communication and fleeting knowledge networks that form around the development of a set of dependent requirements in terms of membership and the nature of interaction. This characterization aims at revealing hidden requirements-driven collaboration patterns, which are recurring repetitions of the same collaboration behavior across the social networks of dependent requirements. Moreover, it aims at bringing insights about how cross-functional software teams manage requirements evolution and change information, and what gaps in communication and fleeting knowledge need to be identified to avoid loss of critical knowledge within these networks. In this research fleeting knowledge is conceptualized as the awareness that members have of what others are doing that is related to one's work, also named *current awareness* [48].

Research question 3: *What structures do communication and fleeting knowledge networks have in requirements-driven collaboration?*

Actual interaction among team members determines a networks structure, which emerges as the members perform the project tasks. Researchers in the area of social networks have found that network structures can be used to explain collaboration behavior [3]. In communication networks the structure can be recognized as the patterned flows of information exchanged among team members [111]. For example, a social network representing collaboration around a requirement that consists of a structure centralized around certain members may suggest that there are key people

responsible for distributing knowledge to others. These members may be experts about the requirements or senior developers who are familiar with the product under development. In addressing this question I intend to examine which structures the communication and fleeting knowledge networks have in requirements-driven collaboration aiming to reveal who holds knowledge and who actively distributes it to colleagues. Moreover, it also aims at identifying who is most aware of others or who others are most aware of in order to understand communication breakdowns or misunderstandings that happen because of lack of awareness.

Research question 4: *How do team members exchange information and share knowledge relevant to performing work driven by requirements?*

In studying collaboration driven by requirements, I posit that team members are specialists. Given each project member's different role in the project, a member has specific knowledge about his respective requirement. For example, the requirements analyst may have different knowledge about the requirement than the developer or the tester. The study of how information is exchanged among these members who possess specific knowledge about requirements should reveal insights about patterns of knowledge distribution within the team. In addition, the study of communication patterns should reveal dynamics of collaboration among people that need to coordinate their requirements-related work. Knowledge about how team members exchange information and share knowledge can help managers to define strategies to minimize the risk of having knowledge concentrated in few members or not being distributed to certain people, to better fit the assignment of members with certain knowledge to work on certain requirements, and to improve the use of tools that support knowledge exchange among team members, specially when the team is physically distributed. By posing this question I aim to identify situations that indicate particular patterns of information exchange and knowledge sharing among those involved in requirements-driven collaboration.

Research question 5: *To what extent are requirements-driven collaboration patterns aligned with coordination needs informed by requirements dependencies? Moreover, to what extent do requirements-driven collaboration patterns follow the organizational structure?*

Requirements dependencies drive the need to coordinate work activities. Since requirements are usually volatile and changes may affect requirements dependencies,

actual coordination among team members may not correspond to the coordination needs created by the project requirements dependencies. Therefore, the goal of this question is to examine the extent of the alignment between coordination needs and the team's ability to coordinate work aiming to inform which coordination needs were not satisfied. In addition, formal organizational structures often define communication channels among team members. Requirements-driven collaboration patterns may be affected by these imposed organizational structures. Thus, this second part of question 5 aims to identify to what extent collaboration driven by requirements-related activities follows the communication channels imposed by the organization structure. Awareness of the alignment between the project's coordination needs and the organization structure can help managers to identify whether changes in the organization structure are necessary to smooth requirements-driven collaboration among team members, or when collaboration was not accomplished because of other reasons.

1.1.3 The Investigative Approach Used in this Dissertation

To study requirements-driven collaboration, I propose an approach that uses concepts and techniques from social network analysis [135] to obtain insights about the communication and fleeting knowledge patterns of those involved in requirements-driven collaboration. This approach is based on a structure that focuses on the requirement as the unit of work around which collaboration occurs. I term this structure a requirements-centric team. A requirements-centric team (RCT) is a cross-functional group whose members' work activities are related to one or more interrelated requirements, as well as downstream artifacts such as design, code and tests. By *related to* I consider relationships such as *assigned to* and *communicating about*.

To analyze the collaboration within RCTs, I define a requirements-centric social network. A requirements-centric social network is a social network [135] that represents the members (also called actors) and relationships (also called ties) in a requirements-centric team. The actors in a requirements-centric social network are among the members of the requirements-centric team, and the ties in the network are representations of different relationships during these members collaboration. For example, a tie can represent project members' requirements-related communication, assignment to work on the same requirements, contributions to the development of a requirement, or awareness of others' requirements-related work.

Having defined requirements-centric social networks, a number of social network

analysis measures are defined as mechanisms to the study of properties of these networks. Insights from the application of these measures allow one to identify requirements-driven collaboration patterns.

1.1.4 An Empirical Investigation of Requirements-Driven Collaboration

To empirically examine collaboration driven by requirements and identify the applicability of the proposed framework, I conducted a multiple exploratory case study. I investigated two projects in the Brazilian subsidiary of a large manufacturing American company, which I will call ORG for confidentiality reasons. ORG has offices all over the world, including development centers in the US, Brazil, and India. ORG assembles and ships its products worldwide, and has an extensive I/T department to support its internal processes.

The *Shipping System* project updates and maintains an internal mature software product used by ORG to support its shipping process. The *Support Applications* project enhances and maintains a group of internal software products used in ORG by product management and sales. The two projects were selected mainly according to availability and easiness to reach the remote members located in US.

I visited the project teams on site multiple times, spending three initial full months observing each project team in its work environment and periodically returning to collect feedback on preliminary findings and additional data to complement the understanding of the project contexts. At the end of the initial observation period, both projects had released the new version of the software into the production environment and were providing the product probation support time which consisted of addressing eventual end-users bug fix reports. I used document inspection, questionnaire, work diaries, semi-structured interviews, and observation for data collection. Data analysis was conducted using the requirements-driven collaboration approach I proposed which utilizes concepts and measures from social network analysis, as well as interview data analysis and descriptive statistics.

1.2 Contributions

This dissertation brings several theoretical and empirical contributions mainly to the Requirements Engineering field by answering the posed research questions. In

terms of theoretical contributions, this dissertation proposes an *empirically-informed framework* for investigating requirements-driven collaboration. This framework was incrementally developed throughout the research, first informed by literature review and then empirically-informed through its application in a multiple case study of requirements-driven collaboration. The framework consists of an approach based on a structure that focuses on a requirement as the unit of work around which collaboration occurs. This structure represents a cross-functional team whose members' work activities are related to one or more interrelated requirements, as well as downstream artifacts. It also defines the requirements-centric social network concept, which represents members and relationships in a requirements-centric team. Moreover, the framework outlines a number of social network analysis measures thoroughly selected as mechanisms for studying collaboration driven by requirements. These measures can be one-by-one applied to the requirements-centric social networks in order to identify requirements-driven collaboration patterns, such as which team members are involved in negotiating requirements, how notification of changes are propagated to those affected by them, and who holds knowledge about the requirements.

Another theoretical contribution is the proposed *Role-based socio-technical congruence measure*. Based on the current socio-technical congruence measure defined by Cataldo et al. [24], the Role-based measure aims at examining requirements-driven collaboration in the presence of a formal organization structure represented by the pairs of roles that are allowed to directly communicate with each ther. In other words, the proposed Role-based measure indicates the extent that actual coordination attends coordination needs that are aligned to the formal imposed communication channels. It also indicates which missing communication between a pair of people is between roles that should not directly communicate with each other, and thus coordination needs may not have to be satisfied. These missing communication instances are named *false gaps*. Moreover, it identifies communication that takes place between pair of roles that should not directly communicate with each other, named *backchannel communication*. Managers can benefit from the awareness that backchannel communication takes place in their projects, and use the detailed information about this type of communication to align or to supplement the formal organization structure with the project coordination needs.

This dissertation also brings significant empirical contributions. A substantial initial body of knowledge, grounded in empirical evidence, about requirements-driven collaboration was developed. The multiple exploratory case study conducted re-

vealed contrasting requirements-driven collaboration patterns for the two projects. In the effort to characterize the communication and fleeting knowledge networks in requirements-driven collaboration, the application social network measures proposed in the framework revealed that requirements-centric social networks are dynamic and included important cross-site and cross-team interactions. These networks have decentralized structures, meaning that knowledge is not hold by few members and is well distributed among team members. In terms of patterns of information exchange and knowledge sharing, the application of the social network measures uncovered that there are members brokering information between two otherwise unconnected colleagues. In the context of requirements engineering, this indicates that these members whose communication was brokered did not use and possible had no direct way of communicating requirements information. Misunderstandings or information loss could occur in these mediated interactions.

The application of the current socio-technical congruence measure [24] and of the proposed Role-based measure revealed that requirements-driven collaboration patterns were aligned with coordination needs and with the organization structure for the project where the team had previous knowledge about the software and members are familiar with each other. In contrast, for the other project where the team was new to the application and as a group itself, the requirements-driven collaboration patterns were mostly misaligned with the teams' coordination needs and the imposed organizational structure. The Role-base measure identified that backchannel communication did took place in both projects in different proportions, meaning that to a certain extent both teams overlooked the imposed communication channels in order to attend coordination needs created by the requirements dependencies.

The contributions in this dissertation bring implications for requirements engineering and for the study of collaboration is software projects. By providing researchers and practitioners with a set of measures to study and evidence about, the framework offers a mechanism to examine fine-grained details about requirements-driven collaboration. The insights obtained can be used to reason about how tools and processes can be improved to better support collaboration throughout the development life-cycle.

1.3 Structure of this Dissertation

The remaining of this dissertation is organized as follows.

Chapter 2. Literature Review This chapters presents literature on collaboration in software engineering emphasizing needs and challenges imposed by requirements engineering and cross-functional teams, specially those who work in globally-distributed settings. In addition, it introduces social networks concepts and how they have been used to investigate collaboration in software engineering. This chapter also presents the socio-technical concept as adopted in the software engineering community in the study of collaboration and its incipient related work.

Chapter 3. Research Methodology This chapter presents the adopted research process. More specifically, it briefly introduces the investigative approach defined to investigate requirements-driven collaboration and presents case study as the strategy used to empirically examine collaboration driven by requirements in two projects, describes the criteria used to select the investigated projects, and briefly presents the data collection and analysis methods adopted as well as the research validation process.

Chapter 4. A Framework for Studying Requirements-Driven Collaboration Seeded in literature review, the proposed framework for studying requirements-driven collaboration is introduced in this chapter. The requirements-centric teams and the requirements-centric social networks concepts are defined here, and the initial set of social network analysis measures used as mechanisms to explore requirements-driven collaboration is also presented.

Chapter 5. Case Study Data Collection and Analysis This chapter discusses in details the data collection and analysis methods and processes used in the investigation of the multiple exploratory case study. The data collected for each project and steps taken to analyze the data are also presented here.

Chapter 6. The Shipping System Project This chapter presents the empirical case study named *Shipping System*, shortened to SHIP, and discusses the insights obtained about requirements-driven collaboration from the detailed examination of the team behavior. First, the SHIP project setting is presented in details. Then, the analysis of the data collected is presented and discussed in such a way that the research questions are addressed.

Chapter 7. The Support Applications Project Similarly, this chapter presents the empirical case study named *Support Applications*, APP in short, and discusses the insights this study brings in light of requirements-driven collaboration. The structure of Chapter 6 is followed. First, the APP project setting is described followed by the data analysis and discussion of results.

Chapter 8. Revisiting the Research Questions This chapter revisits the research questions by contrasting and highlighting the main insights from the multiple exploratory case study aiming to summarize how this research furthers the understanding of requirements-driven collaboration. The final set of social network analysis measures defined as mechanisms to explore requirements-driven collaboration in the proposed framework is presented.

Chapter 9. Final Considerations This chapter presents the research validation process adopted throughout the research, discusses the contributions of this research for furthering the understanding of requirements-driven collaboration, and disclose the threats to the validity of the contributions. The chapter concludes this dissertation with suggestions for future work.

1.4 How to Read this Dissertation

Each and every chapter in a dissertation is written aiming to support the understanding of the research I have conducted. The reading of the entire dissertation in the order of presentation of the chapters is expected but it is not mandatory. To support the reader, I suggest an alternative reading order which I believe would not compromise the comprehension of the work here presented. This alternative order applies for readers who are assumed to have previous knowledge on social network analysis and want to prioritize the reading of the research contributions.

Chapter 1 (Introduction) sets up the context of this research. Motivation for investigating requirements-driven collaboration and research goal and questions are presented here. Once the reader has a broad view of the research, he may want to first read about how the research questions were answered and about the research contributions itself. Thus, the reader should move from Chapter 1 to reading Chapter 8 (Revisiting research questions), and Chapter 9 (Final consideration). Next, for details of how the research questions were answered, I suggest the reader to move to Chap-

ter 4 (Framework), Chapter 6 (SHIP case study), and Chapter 7 (APP case study) in this order. Chapter 3 (Methodology) and Chapter 5 (Case study data collection and analysis) provide details about the methodology used to examine the research questions. Last, Chapter 2 (Literature) will serve the purpose of defining concepts used in this research and introducing related work that inspired and motivated this research. Appendices consist of relevant information for readers who would like to conduct similar research (Appendix A - Questionnaire sample), for those who are interested in specifics about each project (Appendix B - SHIP project, and Appendix C - APP project), and for those that would like to use the framework themselves (Appendix D - the complete framework).

Chapter 2

Literature Review

In this chapter literature review related to collaboration and organizational structures as determinants of communication behavior. It also presents literature about social network analysis and socio-technical congruence as approaches to investigate collaboration. Concepts associated to these topics are defined and related work is described.

2.1 Requirements Engineering and Its Importance to Software Development

The success of a software system depends on how well it fits the needs of its stakeholders, and its environment [101] [104]. Software requirements comprise these needs, and requirements engineering is the process by which the requirements are determined [27]. A requirement is a description of how the software system should behave, a property or attribute exhibited by a system or a system component to solve a problem or achieve an objective [121].

The requirements engineering process involves understanding the needs of users, customers, and other stakeholders; understanding the contexts in which the to-be-developed software will be used; modeling, analyzing, negotiating, and documenting the stakeholders requirements; validating that the documented requirements match the negotiated requirements; and managing requirements evolution [27]. Changes to requirements require not only management, but also impact analysis to understand whether the changes affect the current system requirements, specification, and design [101].

Successful requirements engineering is important since it establishes what needs to be designed [92]. A poor requirements engineering and management may result in project failure, misunderstandings and conflicts between stakeholders and project team, rework, and high number of defects. Thus, improvements in the requirements engineering process have the potential to reduce development costs and development time, and to increase software quality [36] [50].

In addition, requirements are used in the following phases of the life-cycle to guide the project activities. The requirements are then used by the software team to develop, to test, and to deploy the software system in the user environment. Team members working in different phases of the development life-cycle and belonging to different organizational functions need to communicate and to share a common understanding about the requirements in order to successfully develop the requirements, thus attending the stakeholders needs.

2.2 A Requirements Engineering Perspective on Collaboration in Software Projects

Requirements engineering drives software life-cycles from the elicitation phase down to the analysis, implementation and test phases [6]. As such, it involves collaboration among large, often geographically distributed cross-functional teams comprised of requirements analysts, software architects, developers, and testers. This collaboration is driven by coordination needs in software development and relies on communication.

Coordination, generally defined as the act of managing interdependencies between activities [95], is a critical aspect in every activity related to a requirement's analysis, implementation or testing. Team members coordinate to establish a common understanding and to share knowledge about the work to be done. Since requirements are volatile [36] [18], ongoing coordination is necessary to manage interdependencies with those working on the artifacts related to a changed requirement. Changes in one requirement need to be propagated to those who work on dependent requirements and related downstream artifacts. Ineffective coordination with those who work on dependencies may result in failures [38].

Team members use diverse coordination mechanisms. Espinosa and colleagues [52] categorized these mechanisms as **mechanic** via task programming mechanisms and **organic** through team communication. Coordination of repetitive and routine

aspects of the task can be achieved mechanistically using tools, schedules, plans, specifications, etc. A configuration management system, which enables developers to work simultaneously in the same code and to know who is working on this code, is an example of mechanistical coordination. Aspects of a task that cannot be supported mechanistically, such as deciding how to proceed when a deadline is missed or asking for a task clarification, need organic coordination also known as *coordination by feedback*. This type of coordination is done through communication [52]. In addition to communication, the knowledge developed about the project tasks and the team also help team members to coordinate [79].

Communication is important in requirement engineering because it facilitates the negotiation process among different stakeholders, allows team members who are responsible for defining the requirements to clarify information for other members [101] and to share knowledge necessary to capture, define, design, and implement the software requirements [92]. Poor communication will result in failures to perform these activities, in misunderstandings, and in conflicts [31]. As a consequence, project performance and customer satisfaction are impacted. Both informal and formal communication are valuable mechanisms for achieving coordination in software development [84].

Team members' common ground, knowledge of other members working in the same task and the task domain, familiarity with the task and other members, and awareness of who is around and who has done what recently are few examples of how knowledge can help team members to coordinate more effectively. For instance, requirements analysts and architecture designers have to share a common ground about the requirements in order to the requirements analysts be able to clarify requirements to designers, and designers be able to understand and design a software architecture that addresses the defined project requirements.

Aspects pertaining to the *knowledge* that supports coordination are categorized as long-term knowledge, and fleeting knowledge or awareness [52]. **Long-term knowledge** is acquired over time and is more permanent. Knowledge of the process used by a team to elicitate and negotiate requirements with customers and stakeholders, knowing who has expertise in the team to clarify the meaning of a certain requirement, or acquiring knowledge with the customer about the requirement in case no member in the team has expertise to clarify it, are examples of this type of knowledge. Kraut and Streeter [84] found that knowing who to ask for help facilitates coordination, and is beneficial for the software project success. Long-term knowledge helps

members coordinate because it helps individuals develop more accurate explanations and expectations about tasks events and members behaviors [81].

In contrast, the **fleeting knowledge** or **awareness** is situational and temporarily relevant [52]. For example, a tester knows which requirements analyst was responsible for specifying the requirement because he asked clarifications about the requirement behavior in order to write test cases for that requirement. A few weeks later, this knowledge may no longer be useful. Awareness is knowledge about the state of a particular situation, object, or environment [1]. It is also defined as an understanding of the activities of others, which provides a context for your own activity [46].

Awareness is important for coordination of tasks that contain interdependent activities, because it helps members shift from individual to shared activities easily, and because members have a better understanding of the sequence and timing of things and temporal boundaries of their actions [64]. Awareness is the beginning of self-understanding and it reduces the effort needed to coordinate tasks and to anticipate other team members' actions [46] [65]. Thus, awareness is important for cross-functional teams that are brought together to develop requirements in a project and that have to coordinate work to achieve the project's goals.

There are several types of awareness. The most common among researchers studying collaboration and coordination are as follows. **Task awareness** is a member's up-to-the-minute knowledge of what is going on in the task in areas that affect that member's work [52]. Ehrlich and Chang [48] name this knowledge *current awareness*. This definition is similar in concept to Chen and Gaines' concept of *chronological awareness* [26], which is knowledge of recent task activities (e.g., knowing who did what recently, who is behind schedule). By knowing the task activities of others, team members can coordinate their work more effectively [52].

Presence awareness is the knowledge of which team members are around, where and when, as relevant for the task. When members have tight dependencies with other members, it is important to be able to find the right people when you need them, or at least to know when and if they are around [52]. This awareness is also known as workspace awareness. Gutwin and Greenberg [64] defined *workspace awareness* as the understanding of people in the workspace, rather than just of the workspace itself. This awareness is more hard to be achieved in globally distributed projects since team members do not have mechanisms to identify where their colleagues are currently located. For example, the use of clues such as a jacket hanging on a chair is a privilege for collocated team members.

The knowledge about the team members' professional background and how they can help you in your work is defined by Ehrlich and Chang [48] as **general awareness**. This knowledge is important to facilitate expertise seeking and to help a team member to complete a task. **Change awareness** is the ability of individuals to track the changes made on a piece of work, such as a requirements specification document or source code that implements certain requirement, or on decisions took on the project, such as the delivery date of the software system to the customer [128]. This awareness is significant in requirements engineering since changes on requirements have to be propagated to team members involved in the development of that requirement to avoid misunderstandings or rework later on. Specifically, knowledge of changes to requirements that are dependent on other requirements is needed to ensure that team members are performing their work based on the current version of the requirement specification.

Although research has pointed out the importance of communication and knowledge for coordination [52], communication is a recurring problem in requirements engineering [6] [31] [40]. This problem occurs because communication involves a diverse range of team members who differ on levels of background and knowledge about the project requirements. Requirements analysts have to gather and negotiate requirements with customers, who understand about the business process that will be supported by the software system, and help architecture designers to understand the requirements. Requirement analysts also support developers and testers on the understanding of how to code and test the requirements. Software architects have knowledge about how to define a solution that will implement the requirements. Thus, these team members that belong to different organizational functions have to establish a common ground about the requirements to be able to effectively communicate and exchange knowledge necessary to implement the project requirements.

Distributing the software development process is a practice that has became common in the last years [69]. This distribution happens for diverse reasons, among others because of size and complexity of software systems that require larger teams of qualified people to work on them. Coordination issues are exacerbated when the development is globally distributed. Distance affects team members ability to meet face-to-face and to discuss project issues. Not only the geographical distance affects coordination, but also the organizational dispersion, inherent in any software team. Members of cross-functional software teams usually belong to different functional and organizational groups. Different functional and organizational groups in general have

different work policies and practices, which will imply in different ways to communicate and exchange knowledge necessary for developing work driven by requirements.

2.3 Organizational Structures and Communication Channels

An **organizational** structure is mainly a hierarchical concept of subordination of entities that are expected to collaborate and contribute to serve one common aim. Traditionally, an hierarchical organization is designed to thrive upon the division of labor, workflow, and unity of command [109]. That is, in order to increase efficiency and control, employees work in functional units or departments; and each employee specializes in a few tasks [70] differing in the expertise, knowledge, and information that each one brings to the group [74]. The structure of an organization will determine the modes in which it operates and performs. Hence, an effective organizational structure shall facilitate working relationships between various entities in the organization and may improve the working efficiency within the organizational functions.

Organization literature claims that information should flow along the lines of the prescribed organizational structure [55]. However, it is known that the organization structure may create obstacles to the movement of information by blocking flows and by promoting overload of information in certain paths. People usually find ways to overcome these obstacles such as disregarding the organization structure and establishing informal relationships across functions to faster accomplish their tasks. These so called "informal organization networks" represent informal group structures formed to acquire information or to get help from other colleagues aiming to accomplish the work that needs to be done. These informal group structures can cut through formal reporting procedures to jump start stalled initiatives and meet extraordinary deadlines [82].

An informal group structure can provide communication links that reinforce and add to those networks of formal organization charts due to the ability of those involved to collaborate, to exchange knowledge, and to pool expertise across hierarchical boundaries [33]. These informal links characterize backchannel communication [55]. Often these informal communication channels or backchannel communication increase group performance. Past research has argued that group performance is affected by the distribution of knowledge among members of a group. Wherever information is

broadly distributed across group members, these members share common information and conceptualization, and this in turn facilitates group performance. In groups where pockets of unique knowledge are concentrated in specialists, such as cross-functional software teams, critical information possessed by certain members may never be shared or retrieved by the group, and this can diminish group performance [113].

However, one factor that may modify the effect of distribution of knowledge is the group structure. Studies show that the group structure can either constrain or facilitate knowledge and information flow within a group. Somewhat centralized information flow, for example through brokers, would represent a hierarchical communication structure. Similarly, flat networks or decentralized groups provide opportunities for task-oriented communication and information exchange [124] [7]. Such a dense communication structure enables the free flow of information and thus broader distribution of knowledge. The performance in groups of generalists with broadly distributed knowledge does not seem to be affected by the group structure. Whereas centralized groups of specialists were outperformed by decentralized groups [113].

On the other hand, group structure seems to play an important role when one considers knowledge dissemination across teams. Past studies of information flow in distributed groups (e.g., [71]) found that distributed teams with more centralized structures experienced fewer coordination problems, whereas dense communication was associated with more coordination problems. Some hierarchy caused by particular team members acting as point people (i.e., brokers) through whom much of the cross-team communication flowed, was identified as critical to ensuring coordination across teams [71]. Thus, I am interested in identifying which social network structures are formed during requirements-driven collaboration. These structures are expected to influence knowledge exchange and information flow within requirements-centric social networks of dependent requirements.

2.4 Social Network Concepts and Measures and Their Use in the Study of Collaboration

People establish relationships with others all the time. Software developers are not an exception. When developers are allocated to work on a same requirement and frequently meet to discuss technical-related issues and development progress, they

establish a stronger working relationship to each other than to those in the team that they are not in contact with. As time goes by in a software project, team members also develop a trusting relationship on each other which can help them to open substantive and influential information exchange channels [5]. Social networks analysis provides techniques to examine the structure of such social relationships in a group to uncover patterns of behavior and interaction among people [99]. It is particularly useful to unveil informal relationships [34], such as those between team members who play roles that should not communicate with each other because of imposed organizational restrictions.

Social network analysis is founded on the concept of a **social network**, which is a social structure made up of individuals (or organizations) called nodes, which are tied or connected by one or more specific types of relationships [99], such as friendship, trust, dislike, knowledge, communication, or awareness. Therefore, in social network analysis **nodes** are the individual actors within the networks, and **ties** represent relationships between the actors. Social network measures are used to examine different aspects of the resulting graph-based structures that represent the social relationships of one's interest.

A finite set of actors on which ties are to be measured is called **group**. The collection of ties of a specific kind among members of a group is called a **relation**. For example, a social network can represent a friendship or a communication relationship. A social network consists then of a finite set of actors and the relation or relations defined on them [99]. Although social network data require measurements on ties among actors, called **relational data**, characteristics of the actors may also be collected and analyzed [117]. Such characteristics are named **attributes** and can identify the actor's educational background, the time the actor is working in a company, his job position and office location, among others.

Many kinds of network analysis are concerned with understanding ties among pairs. These analyses use the **dyad** unit of analysis, which consists of a pair of actors and their possible relationships [135], such as whether or not ties are reciprocal. Another set of network analysis is focused on the study of a **subgroup** and all ties among them while some measures concern to the examination of the **entire network**.

I will use a running example to illustrate social network terminology and measures of interest. These measures are listed in Table 2.1. As a fictional example consider the following scenario. A software team composed of 12 members is organized as follows. Andrew, Bob, Charles, David, and Emma are testers responsible for verifying and for

Table 2.1: Social network measures and concepts introduced in this section

Sociogram	Clique
Size	Component
Density	Reachability
Ties statistics	Cutpoint
Centralization	Degree centrality
Core-periphery	Brokerage
Ties reciprocity	Socio-technical congruence

validating what the developers have implemented. The developers are who are Fynn, Greg, Hannah, Iris, John, Kevin, and Lucas. Two-thirds of the team is collocated in the West Coast of Canada, however Bob, David, John, and Kevin have their offices located in Ireland. The software team consists of members with different levels of experience working with the product under development. Senior members Andrew, Bob, Fynn, and John are working in the project for more than 18 months while the newcomers Emma, Greg, and Hannah have joined the project less than 6 months ago. The remaining members, Charles, David, Iris, Kevin, and Lucas, are working in the project between 6 and 12 months. No one has worked for more than 12 months and less than 18 months.

Table 2.2 indicates those members who frequently (over 3 times a day) communicate in the project. This table can be viewed as a **sociomatrix** where each name of a row (or column) represents an *actor* or a *node* in the social network. Note that I shortened to the first letter the actor names in the columns (e.g., Andrew is indicated "A", Bob is indicated "B", and so on). The value in each cell ij represents that there is a relationship, value 1, or absence of it, value 0, between actors i and j. The presence of a relationship between actors i and j defines a *directional tie* between these actors in the social network. Note that in this illustration there is a tie between Lucas and John, but the inverse is absent. The social network in this running example has *directional ties*, when distinction of who initiated the relationship is made. Therefore, one must read the matrix as follows: actor a in row i frequently communicates with actor b in column j if the value in cell ij is 1. Otherwise, a and b do not frequently communicate. Note that self-relationships are not considered.

To obtain overall insights about the characteristics of the social network, the social data matrix can be displayed as a sociogram. A **sociogram** is a picture in which actors are represented as points in two-dimensional space, and relationships among pairs of actors are represented by lines linking the corresponding points [100].

Table 2.2: Frequent communication interactions between team members of the running example

	A	B	C	D	E	F	G	H	I	J	K	L
Andrew		1	1	1	1	1	0	0	0	0	0	0
Bob	1		1	1	0	0	0	0	0	0	0	0
Charles	1	1		1	0	0	0	0	0	0	0	0
David	1	1	1		1	0	0	0	0	0	0	0
Emma	1	0	0	1		0	0	0	0	0	0	0
Fynn	1	0	0	0	0		1	1	1	1	1	1
Greg	0	0	0	0	0	0		0	0	0	0	0
Hannah	0	0	0	0	0	0	0		0	0	0	0
Iris	0	0	0	0	0	1	0	0		1	1	0
John	0	0	0	0	0	1	0	0	1		1	0
Kevin	0	0	0	0	0	1	0	0	1	1		1
Lucas	0	0	0	0	0	1	0	0	0	1	1	

The lines can be *reciprocal* (two-ways), *directional*, and *valued* (when a connection represents the intensity or magnitude of a relationship). Figure 2.1 presents the sociogram of the illustrative running example. Note that not only the relationships between actors are indicated but also the actor attributes are visually represented. For example, the *actor shape* indicates the role the actor plays in the project (square for tester and circle for developer). The *actor color* indicates the actor office location (black for the West Coast of Canada and grey for Ireland). The *actor size* finally indicates the time the actor is working in the project (small for 1 to 6 months, medium for 6 to 12 months, and large for over 18 months). From the sociogram one can say that both roles are physically distributed and have newcomer members. It is important to observe that large social networks may not benefit as much from the sociogram. However, by combining actor attributes to relational data one can visualize subsets of the network which may help one to overcome the challenge of one trying to obtain insights from the visualization of an entire large social network at once.

In literature, Damian et al. [39] investigated how distributed developers collaborate over long distances and maintain awareness of others. Sociograms were used to examine who communicated with whom. They found that social networks revolving around particular units of work are dynamic throughout development, and therefore awareness needs to be maintained in organic infrastructures of work. Ohira and colleagues [103] used the visualization of social networks and collaborative filtering to

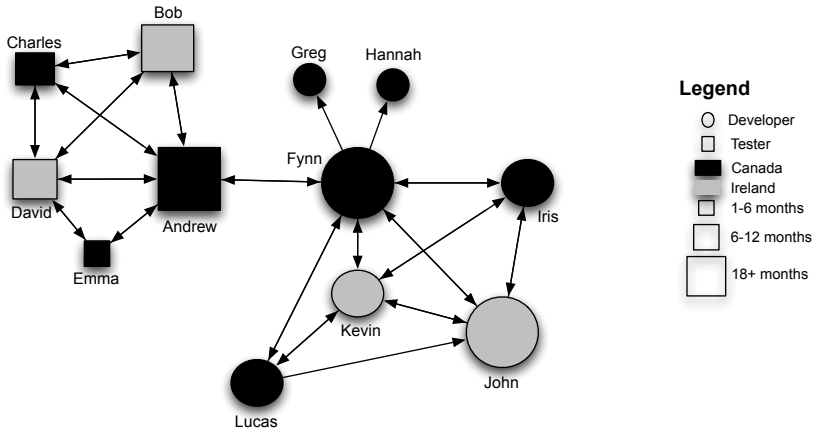


Figure 2.1: Sociogram of the illustrative running example

support the identification of experts across open source projects. de Souza and colleagues [42] developed Argur, a tool to explore the structure of a system through the visualization of source-code dependencies. The tool builds socio-technical networks that highlights links between people through the source code and vice-versa.

Network size is the number of actors in the social network. In the example, the social network that represents communication in the project has size 12. In other words, there are 12 team members frequently communicating with each other in this project. Note that the network size can be smaller or larger than the team size. For example, it is possible that a certain team member did not frequently communicate with any of his colleagues, thus the network size would be 11. It is also possible that external people frequently communicate with team members. In this case, assuming that the 12 team members frequently communicate with each other and that there are 2 additional people communicating with them, then the network size would be 14.

Carroll and Teo [22] found that the size of discussion networks of nonmanagers members are in average 37% larger than networks of managers. Herbsleb and Mockus [68] found that distributed social networks are significantly smaller than same-site social networks. The mean network size for local people that one typically interact with at work during the course of a week was 16 and for remote people was 4.9. Although the proportion is less significant, social networks of communication with local (mean equal 7.6) and distributed (mean equal 4.8) colleagues followed the same pattern.

Network density is defined as the proportion of ties that exist in the network out of the total possible ties. Network density varies from 0 to 1, where 0 indicates that there are no ties present and 1 indicates a complete network where all possible ties are present. In the example, there are 37 ties out of the 132 (12 times 11) possible ties, thus the network density is of 0.28. This low density indicates that of the possible communications that could have taken place in the project less than one-third in fact took place.

Network density has been studied in literature in relationship to predicting team performance [9] and to coordination ease in distributed teams [71]. In Hinds and McGrath study [71], they found that geographic distribution is associated with less dense work ties and less dense information sharing, suggesting that social ties are not particularly important in distributed as compared with collocated teams as a means of coordinating work and improving performance. Hinds and McGrath also used the network centralization and the E-I index measures to investigate the correlation of network structure with coordination ease. They found that although flatter network form is associated with more smooth coordination in collocated teams, the opposite is true for geographically distributed teams. An informal hierarchical structure was associated with more smooth coordination in distributed teams.

Another way to reveal characteristics of a social network is to provide **descriptive statistics about the relationships** established by actors based on their attributes. A subset of statistics for the illustrative example are: (1) the team consists of 5 testers (3 located in Canada and 2 in Ireland) and 7 developers (5 in Canada and 2 in Ireland); (2) 37 instances of communication took place in this project, distributed as follows: 19 within developers, 16 within testers, and 2 cross-teams; and (3) the 37 instances of communication are distributed by location as follows: there are 12 ties within the Canadian site, 4 within the Irish site, and 21 cross-sites. These statistics reveal that: (1) two-thirds of the project team is collocated in Canada, which may facilitate communication among team members; (2) developers and testers are loosely connected, which might impact the team's ability to exchange information; and (3) there is little communication between the Irish members, which suggests that the Irish members may not be aware of each others' work.

In literature, by counting up the number of ties within and cross-sites, Herbsleb and Mockus [68] found that there is much more frequent communication with local colleagues in a distributed project. Respondents communicate with the majority of their local colleagues at least once a day while they communicate with the majority

of colleagues at other sites less than once a week.

In addition to the network overall characteristics, one can study its structure which is defined as the arrangement of the set of ties that link the actors in the network [111]. Some measures to reveal the network structure are as follows. **Network centralization** is a measure proposed by Freeman [58] to quantify the difference between the number of ties for each node divided by maximum possible sum of differences. A centralized network structure will have many of its ties dispersed around one or a few nodes, while a decentralized network structure is one in which there is little variation between the number of ties each node possesses. The network centralization index ranges from 0 to 1. It reaches its maximum value of 1 (fully centralized) when one actor chooses all other actors and the other actors interact only with this central actor. The index attains its minimum value of 0 (fully decentralized) when the number of ties for each actor are equal. Networks that are intermediated of these two have indices between 0 and 1, indicating varying amounts of centralization of degree. The social network in the running example has centralization of 0.39 for its ties sent out to others, which is called **outdegree centralization**, indicating that the *out* ties are not as much centralized in few actors. In addition, it has centralization of 0.19 for its ties received from others, which is called **indegree centralization**, indicating that the *in* ties are even less centralized in few actors than the *out* ties. In other words, the *in* ties are more equally distributed among the actors.

Ahuja and Carley [3] used network centralization to empirically examine the communication structure of a virtual design organization and found evidence of a highly centralized structure where interactions are mediated by a supervisor. Their finding contradicts previous claims that virtual organizations tend to be decentralized. Tsai' study [129] revealed that a formal hierarchical structure in the form of centralization has a significant negative effect on knowledge sharing among organizational units. By applying the network centralization and the core-periphery test, Long and Siau [90] examined the dynamics of social network structures in multiple open source software teams and found that the interaction pattern of these projects evolves from a single hub at the beginning to a core-periphery model as the projects move forward. In a follow-up study Long and Siau [91] examined the relationship between social network structure (also measured by the network centralization index and the core-periphery fit test) and knowledge sharing in open source software development teams. Results showed that social network structure significantly affects the quantity of knowledge sharing but its does not influence the quality of the knowledge shared.

The **core-periphery** test indicates the extent to which the structure of a network consists of two classes of nodes: a cohesive subnetwork, the *core*, in which actors are connected to each other in some maximal sense; and a class of actors that are more loosely connected to the cohesive subnetwork but lack any maximal cohesion with the core, the *peripheral members* [14]. The core-periphery test indicates a value between 0 and 1, where a high value indicates a strong core-periphery structure. In the example, the test yield a value of 0.47, indicating that a core-periphery structure is not strongly present in this network.

Hinds and McGrath found that communication networks with a strong core-periphery structure lead to less coordination problems than loosely connected networks [71]. In a study of a large industrial distributed cross-functional software team, Wolf and colleagues [137] found that the project's communication network has a large core of members from several functions and locations suggesting that project members actively communicated across functional teams as well as geographical distance. In contrast, Crowston et al. [35] found that the core of open source projects is a small fraction of the total number of contributors. Results of the study of communication networks in a geographically-distributed project by using the core-periphery test showed that over time a group of developers emerge as the liaisons between formal teams and geographical locations. In addition to handling the communication and coordination load across teams and locations, those engineers liaisons contributed the most to the development effort [23].

Another way to study the structure of a network is to examine where in a directed network ties are **reciprocal**. Note that this measure does not assume that the relationship represented in the network is expected to be reciprocal but one must consider this fact in the interpretation of the results. For instance, in a *friendship* relationship reciprocity is expected. However, in an *asks advice from* relationship one might not expect reciprocity. There are four possible relations between a pair of actors A and B: A and B are not connected, A is connected to B, B is connected to A, or A and B are connected to each other. There are two methods to calculate reciprocity, which are: the dyad method and the arc method [66]. In the *dyad method* one calculates the ratio of the number of pairs of actors with a reciprocated tie relative to the number of pairs with any tie between the actors. In the *arc method* one calculates the ratio of the number of ties that are involved in reciprocal relationships relative to the total number of actual ties. The higher the index of reciprocal ties, the more stable or equal the network structure is because the actors mutually exchange information

[110]. Moreover, a high reciprocity index suggests a more horizontal structure while the opposite, a low reciprocity index, suggests a more hierarchical structure [66]. In the running example the dyad method yielded an index equal 0.85, and the arc-based method is equal 0.92. In the sociogram (see Figure 2.1) one can see that the pairs of members who are not engaged in reciprocal frequent communication (dyad method) are: Fynn-Greg, Fynn-Hannah, and Lucas-John. Similarly, the ties that do not have a reciprocal pair are those between the 3 listed pairs. These high indexes indicate that the network is stable and that actors equally exchange information.

A **clique** consists of a subset of at least three actors of a network in which every possible pair of actors is directly connected by a tie and there are no other members that are also directly connected to all members of the clique [135]. In directed networks one can identify two types of cliques: a weak and a strong clique. A *weak clique* is when the direction of the tie is disregarded and simply the presence or absence of a relation is considered. In a *strong clique* only reciprocal ties between pairs of actors are considered. This measure provides insights into a group dynamics revealing, for example, the occurrence of informal structures within a network [53]. There are 4 cliques for the social network in the example. The members in these cliques are: Andrew, Bob, Charles, and David (clique 1); Andrew, David, and Emma (clique 2); Fynn, Iris, John, and Kevin (clique 3); and Fynn, John, Kevin, and Lucas (clique 4). Note that cliques 1 and 2 are among testers while cliques 3 and 4 are among developers.

In their study of communication in a software development project, Cain et al. [21] found three large strongly connected cliques consisting of team members developing three major activities: architecture design, code development, and code review. In addition, they found that one of the developers is a cutpoint between the design and the implementation processes.

One may also be interested on how information is exchanged in a social network. Thus, patterns of collaboration within a network is another aspect that one can examine. A set of measures to conduct such examination are presented next. The **reachability** measure defines that an actor is reachable by another actor if exists any set of ties that connects both actors, regardless of how many others fall between them [135]. In order to define reachability in a directed network, one must consider directed paths between actors. For instance, if there is a directed path from actor A to actor B, then actor B is reachable by actor A. It is possible that actor B cannot reach actor A. In contrast, in an undirected network, each pair of actors either are or are

	A	B	C	D	E	F	G	H	I	J	K	L
Andrew		1	1	1	1	1	1	1	1	1	1	1
Bob	1		1	1	1	1	1	1	1	1	1	1
Charles	1	1		1	1	1	1	1	1	1	1	1
David	1	1	1		1	1	1	1	1	1	1	1
Emma	1	1	1	1		1	1	1	1	1	1	1
Fynn	1	1	1	1	1		1	1	1	1	1	1
Greg	0	0	0	0	0	0		0	0	0	0	0
Hannah	0	0	0	0	0	0	0		0	0	0	0
Iris	1	1	1	1	1	1	1	1		1	1	1
John	1	1	1	1	1	1	1	1	1		1	1
Kevin	1	1	1	1	1	1	1	1	1	1		1
Lucas	1	1	1	1	1	1	1	1	1	1	1	

Figure 2.2: Results of the reachability measure for the running example

not reachable to one another. If some actors in a network cannot reach others, there is a potential division in the network and thus information cannot reach everyone. It can also suggest that the team is composed of more than one group. The extent of this division or organization in groups is measured by the *Component* measure presented next. Figure 2.2 presents the results of the reachability measure of the directed social network used as example in this section. For each pair of actors ij the measure indicates 0 if there is no directed path from actor i to actor j, and 1 if there is such a directed path. Self-reachability is not considered. Note that Greg and Hannah are the only members who cannot reach any other member in this communication network.

The **component** test indicates whether a social network is connected. The network is connected if there is a path between every pair of nodes, otherwise it is disconnected. The actors in a disconnected network may be partitioned into two or more subsets in which there are no paths between the actors in different subsets. The connected subsets in a social network are called *components* [135]. For directed networks, one can define two different kinds of components: a *weak component* is a set of actors that are connected, regardless of the direction of ties, and a *strong component* requires that there be a directed path from A to B in order for the two actors be in the same component [66]. Note that the component test differs from the clique test in the sense that the former indicates whether there is a group of people connected to each other (and disconnected from the remaining actors) and the latter indicates whether a subset of actors is completely connected. It was not necessary

Table 2.3: Degree centrality

	Outdegree	Indegree
Andrew	5	5
Bob	3	3
Charles	3	3
David	4	4
Emma	2	2
Fynn	7	5
Greg	0	1
Hannah	0	1
Iris	3	3
John	3	4
Kevin	4	4
Lucas	3	2

to run the weak component test to identify that there is only one weak component in the social network of the example. However, the strong component test identifies 3 strong components in this network, which are divided as follows: Andrew, Bob, Charles, David, Emma, Fynn, Iris, John, Kevin, and Lucas (strong component 1); Greg (strong component 2); and Hannah (strong component 3).

In literature, Reijen and Helms [131] constructed knowledge networks from distinct communication media (email, telephone, and text messaging) aiming to identify how representative knowledge networks can be revealed in organizations. By using some social network analysis measures such as component, density, centralization, and reciprocity to compare each media-based knowledge network with a baseline network captured through in a survey among employees, they found that only the email network is significantly representative for the baselined knowledge network.

Degree centrality indicates the number of ties of an actor and is indicative of activity [59]. In a directed network, the count is made for out-ties (*outdegree*) which are ties from a certain actor to others, and for in-ties (*indegree*) which are ties from others to a certain actor. Table 2.3 shows the out- and indegree of the communication social network. Fynn is the member with the highest outdegree, followed by Andrew. They are also the 2 members with the highest indegree. They are the members who most frequently communicate in the project. Note that two-thirds of the team has equal out- and indegree indices. Fynn, Greg, Hannah, John, and Lucas are the members with an "unequal" communication with and from others.

High degree centrality was found to correlate with the ability of a team member

to coordinate actions of others in a software group. It was also found that highly centralized members coordinate better than others [72]. Individual job performance was found positively related to degree centrality in advice networks [122]. This centrality reflects the individual's involvement in exchanging assistance with co-workers and engaging in mutual problem solving. In their study of a research development virtual team, Ahuja and colleagues [4] found that degree centrality mediates the effects of functional role, status, and communication role on individual performance. In a study of email communication networks in a open source project, the level of activity on the mailing list measured by the degree centrality was found strongly correlated with source code change activity, and to a lesser extent with document change activity. In addition, social network measures such as indegree and outdegree indicated that developers who actually committed changes played much more significant roles in the email community than non-developers [13]. By measuring network centralization and degree centrality in communication social networks, Howison and colleagues [73] found empirical evidence that while change at the center of open source projects is relatively uncommon, participation across the project communities is highly skewed, with many participants appearing for only one period.

In this study of coordination in the Apache open source project, Rosso [112] found that people with high betweenness centrality scores connect people who work on different parts of the software architecture and their role is fundamental for coordination and the flow of information. This finding suggest that central developers may become bottlenecks because of the knowledge they have about the project and the facility they have to find access to information more easily or to find the right person to contact. Networks that contain actors with high betweenness are found vulnerable to having information flows disrupted [34].

Wolf et al. [139] [138] built a model based on the degree centrality and density social network measure to predict the quality of the build outcome, meaning the build pass or fail, in the IBM Rational Jazz project. Lim and colleagues proposed a method called StakeNet [88] to prioritize stakeholders in large projects based on centrality measures. The web-based tool developed to validate the method, called StakeSource [89], identified stakeholders and their roles with high recall and accurately prioritized them. The tool also uncovered critical stakeholders that were overlooked in the past demonstrating its potential in the identification of stakeholders in large projects.

In the field of new product development, Batallas and Yassine [10] used the centrality and brokerage measures to detect central members regarding information exchange

and system integration, who they named *information leaders*. They found that these members are critical for the communication and coordination of tasks among the often large cross-functional team involved in the development of a new product.

Brokerage indicates when an actor, named *broker*, connects two otherwise unconnected actors or subgroups [61]. Brokerage occurs when, in a triad of actors A, B, and C, A has a tie to B, and B has a tie to C, but A has no tie to C. In other words, A needs B to reach C, and therefore B is a broker. The broker actor is in a position to manage or broker information flow. A broker can be problematic if intentionally or unintentionally the actor introduces misunderstandings or limits exchange of information. To calculate brokerage first it is necessary to partition actors into mutually exclusive groups. This partition must be meaningful in accordance to the context of the studied network [10]. There are five kinds of brokers: *coordinator*, when the three actors belong to the same group; *consultant*, when the broker belongs to one group, and the other two belong to a different group; *gatekeeper*, when the source actor belongs to a different group; *representative*, when the destination actor belongs to a different group; and *liaison*, when each actor belongs to a different group. A directed network is assumed. Table 2.4 presents brokerage for the running example. The number in each ij cell of the table indicate how many times member i played the broker role j in the project. For example, one can see that the developer Fynn brokered information among his developer colleagues 11 times. In other words, Fynn coordinated information exchange among his team 11 times in this example. Note that the liaison broker role is omitted from the table since it involves three groups, and the example defines only two groups (developers and testers). Brokerage between office locations and level of expertise (indicated by the time working in the project attributed) could also have been calculated.

Research in global software development have identified that brokers effectively disseminate information between distributed sites when maintaining direct relationships is not practical [71]. They are usually the most knowledgeable members of a team regardless of geographical distance [49]. Milewski et al. [98] found that some locations acting as brokers facilitated collaboration across other locations. They proposed a set of guidelines based on social network theory to help managers establish tactics to deal with brokers locations aiming to increase the collaboration effectiveness.

One can identify which actors are weak points in the network and if they are removed along with their connections, the network would become divided into un-

Table 2.4: Brokerage between project roles (developers and testers)

	Coord.	Gat.	Repr.	Cons.	Total
Andrew	4	4	4	0	12
Bob	0	0	0	0	0
Charles	0	0	0	0	0
David	4	0	0	0	4
Emma	0	0	0	0	0
Fynn	11	6	4	0	21
Greg	0	0	0	0	0
Hannah	0	0	0	0	0
Iris	0	0	0	0	0
John	1	0	0	0	1
Kevin	3	0	0	0	3
Lucas	0	0	0	0	0

connected parts. If there is one such actor, he is called a **cutpoint** [135]. However, if there is a set of actors that fit this criteria, then one would call them a **cutset**. A cutpoint is critical in a group because if he is removed the group will became disconnected and information will not be able to flow between the two disconnected subgroups, at least until the cutpoint actor is replaced. This test reveals how well connected and vulnerable to disruption a network is. Andrew and Fynn consist of the cutset of the illustrative network.

2.5 Socio-Technical Congruence in the Study of Collaboration

Software projects are full of fine-grain dependencies that change on a frequent basis. Conventional coordination mechanisms like formalized procedures or policies [132] have limited applicability in the dynamic software environment. Such dynamic contexts require a mechanism for studying coordination where the project members shifting coordination needs can be easily captured. Socio-technical congruence is an useful approach for this detailed level of study of collaboration in software projects.

Socio-technical congruence (STC) is a measure of the level of "fit" between technical dependencies and coordination in an organization. STC is defined as the match between the coordination needs established by the technical domain and the actual coordination activities carried out by project members [24]. The coordination

needs are acquired by analyzing the assignments of team members to tasks and the technical dependencies among these tasks. The actual coordination is obtained by analyzing the actual communication carried on by team members about the project's tasks. Intuitively, if *Person A* works on one task of the project, and *Person B* works on another task, and the two tasks are dependent of each other, then *Person A* should coordinate with *Person B*. Mismatches between the coordination needs and the actual coordination, also named **congruence gaps** or simply **gaps**, indicate that actual coordination does not reflect the technical dependencies established.

Congruence represents a state, since it captures a particular moment in time, with the actions and needs constrained by the social and technical context at that exact time. Moreover, congruence is dynamic due to the evolving nature of software development. With the passing of time and the actions of individuals during that time, the social and technical structures that define the context of work change. A congruent state in which an organization may find itself at one point in time may therefore no longer be congruent at a later time, and vice versa [116].

Cataldo et al. [24] introduced the concept in the software engineering field, but the original conceptualization of socio-technical congruence is in Conway's Law. Over 40 years ago, Conway stated that the structure of a system mirrors the communication structure of the organization that designed it [30]. He observed that when designing a system, certain alternatives where not pursued by the organization because the necessary communication paths to implement the alternative did not exist. Communication is necessary to coordinate design work that was divided in parts and delegated to subgroups of the organization. Coordination provides the possibility of the separated subgroups to consolidate their work efforts into a unified system design. The roots of Conway's work come from the field of organization theory, where it has long been recognized that organizations should be designed to reflect the nature of the tasks that they perform (e.g., [87]; [19]) [93].

Despite the fact that the STC concept was recently introduced in the software engineering field (in 2006), there is still limited research on the topic. A few empirical studies have shown that congruence possibly has effects on product quality, development progress, and cost [116]. Cataldo et al. [24] found that congruence between coordination needs and coordination activities shortened the development time of modification requests. Similarly, Wagstrom [133] found that the higher the congruence of developers working on a bug, the lower the time to resolve that bug. In their study of a distributed software project, Ehrlich et al. [49] found that an increase

in the number of gaps was associated with an increase in code changes. This increase in the amount of code changes represented a decreasing of performance. When considering the effect of distance in coordination, they found that distributed pairs of developers had more gaps than collocated pairs. Strohmaier et al. [127] studied whether the weighted technical structure of component dependencies correlates with the weighted social structure when maintaining those components. Results suggest that the more a component depends on another one, the more people will be involved in maintaining both components.

The study of socio-technical congruence is dependent on the conceptualizations of coordination needs and actual coordination. Coordination needs may indicate interactions of features at the requirements level, dependencies among components and their interfaces at the architecture and design levels, and functional, data, and semantic dependencies at the code level [116]. For instance, in their study Cataldo et al. [24] defined coordination needs by extracting the dependencies between source code files changed per modification reports. Ehrlich et al [49] also used source code as the unit of analysis of technical dependencies. In association with Kwan and Damian, I have defined requirements that depend on each other as the technical dependency unit of analysis in a previous study [97].

Actual coordination indicates social relationships between two persons in a project that reflect existent coordination behavior. These social relationships can represent any organizational aspect, such as function or geographical location, or communication in various forms [116]. Research on the field have conceptualized actual coordination in several ways. Cataldo et al. [24] captured actual coordination through IRC communication, posted comments in a modification requests-tracking tool, team proximity, and geographical proximity. Ehrlich et al [49] combined self-reported data on previous experience working together with a certain colleague, on frequency of communication, and on frequency of sharing files with a certain colleague to define actual coordination. In my previous work, in association with Kwan and Damian [97], I have also used self-reported communication to define actual coordination. Instead, we collected data per reasons of communication and constructed actual coordination matrices which represent coordination activities that took place for different reasons.

The current model of congruence proposed by Cataldo et al. [24] is based on two implicit assumptions: every dependency is equally important and any coordination is good enough to satisfy a coordination need [85]. Thus, in practical terms, each coordination need and actual coordination is represented by a 0 or 1 value. These

dichotomized values, in theory, fail to reflect the actual coordination nature of a project.

To address this limitation, Kwan et al. [85] proposed a weighted congruence measure that applies weighted edges between 0 and 1 to the represented instances of task assignment, task dependency, and actual coordination. For example, a weighted task assignment definition may represent how many hours one person is supposed to spend on a task, allowing modeling situations where people are partially allocated to tasks. Similarly, actual coordination may indicate the frequency of communication between a pair of people. The calculated weighted coordination needs tells how strong the relations between people are supposed to be, and the weighted actual coordination tells how strong the actual coordination between people are. Their extended socio-technical congruence measurement allows the identification of which gaps are more critical by ranking the gaps according to the amount of lack of coordination calculated between each pair of people.

Previously, Wagstrom suggested the inclusion of weighted edges in the socio-technical congruence calculation by proposing the Weighted Individual Congruence (WIC) metric [133]. WIC differs from Kwan's approach by observing the congruence only for those edges that are incident upon a single person. Wagstrom proposal shifts the socio-technical congruence focus to the individual level. Weighted coordination needs are calculated and compared against actual coordination. WIC results in higher values if communication was observed between actor i and those individuals with whom more coordination requirements exist. In this way, WIC captures not only whether or not communication was present, but also whether or not the communication addresses the coordination needs.

The socio-technical congruence measure proposed by Cataldo et al. is also extended in different ways. Valetto et al. [130] extended Cataldo's work by devising a graph theoretic algorithm that measures congruence in a manner that is mathematically equivalent, but also allows to examine the graph describing the socio-technical system. Such a graph comprises of three types of information: the dependencies among artifacts (technical graph); the interactions among the project stakeholders (social graph); and the work actions carried out by the stakeholders on the artifacts, which connect the above mentioned graphs. The graph is analyzed by an algorithm to rank the relationships in the social graph and to rank congruence gaps. Their approach highlights the critical coordination paths within the development organization.

Wagstrom et al. [134] extended STC to include a decay factor to account for changes in network structure over time. The factor is scaled between 0 and 1, where 0 indicates full decay, relying only the current data, and 1 indicates no decay. The factor is then added to each of the task assignment, task dependency, and actual coordination representations in the current STC measure, and congruence for a certain period between time i and j is calculated as in Cataldo's. The addition of this decay factor allows older task dependencies, task assignments, and observed communications to be slowly removed from the network over time.

Sarma expanded the notion of congruence by creating a measure of expertise congruence that determines how well the expertise allocated to a project matches with the expertise required for the project [115]. Expertise requirements are calculate by computing the relationship between experience that represents past experience of developers working together by a task allocation matrix that represents the current allocation of developers to particular modification requests. Expertise congruence is then determined as the ratio of the number of expertise requirements that match the number of files that need to be modified as indicated in the modification requests matrix, which indicates the actual coordination.

2.5.1 Formal Definition of the Socio-Technical Congruence Measure

The formal STC definition according to Cataldo et al.'s technical and social conceptualizations [24] is as follows. Given m people and n tasks to examine, a **task assignment** matrix contains a 1 in position ij if person i is assigned to task j, and a 0 otherwise. A **tasks dependency** matrix contains a 1 in position ij if task j is dependent on task i, and a 0 otherwise. It is assumed that each task is self-dependent. The **coordination needs** matrix, which indicates if person i should coordinate with person j based on these dependencies is calculated as follows:

$$CN = TA \times TD \times (TA)^t \tag{2.1}$$

where TA is an $m \times n$ tasks assignment matrix that indicates that a person is assigned to a task, and TD is a $n \times n$ task dependency matrix that indicates that a task is dependent on another task. The result is CN, an $m \times m$ coordination needs matrix that indicates who in the project should be coordinating with whom.

Once the coordination needs matrix is obtained, congruence is calculated by comparing this matrix to an **actual coordination** matrix. Examples of actual coordination include verbal or written communication between two people, and two people who may regularly monitor a shared workspace. Such connection between two people is called an **edge**. The actual coordination matrix contains a 1 in position ij if person i coordinated with person j. If an edge exists in the actual coordination matrix, and the same edge exists in the coordination needs matrix, then there is congruence on this edge. The socio-technical congruence index, i.e. the degree of overlap between the two matrices, is calculated as follows:

$$congruence = \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} CN_{ij}AC_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{m} CN_{ij}} \tag{2.2}$$

where AC is an $m \times m$ matrix representing actual coordination. One cannot be congruent with oneself, thus if $i = j$ then ignore the value. The resulting congruence value is a number between 0 and 1, where 0 indicates no congruence, and 1 indicates full congruence. There may be cases where actual coordination involves people who were not assigned to a task. These additional people are ignored in the actual coordination matrix and their appearance in the project do not affect the socio-technical congruence calculation.

To identify whether there is a congruence gap in each ij edge, a **lack-of-coordination matrix** L is calculated by subtracting each value in the actual coordination matrix from the coordination needs matrix as below:

$$g_{ij}(CN_{ij}, AC_{ij}) = \begin{cases} 1, & CN_{ij} - AC_{ij} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, m$, where g_{ij} is the gap value. A value of 1 indicates that there is a gap, and a value of 0 indicates that there is no gap or that there was no coordination needs between i and j in the first place. If $i = j$, then ignore the value.

Chapter 3

Research Methodology

Scientific research is characterized by a systematic and formal way of investigating a certain phenomenon. Sampieri [114] argues that research is the execution of an organized process, which consists of a set of steps. This process is dynamic and requires changes as the research progresses. Inspired on Oates [102] and Crewsell [32] I executed the steps shown in Figure 3.1, which were organized in three main research phases. A certain step may appear in more than one phase.

Phase 1 consisted of activities related to the conceptualization of the research. Phase 2 focused on furthering the development of a framework to investigate requirements-

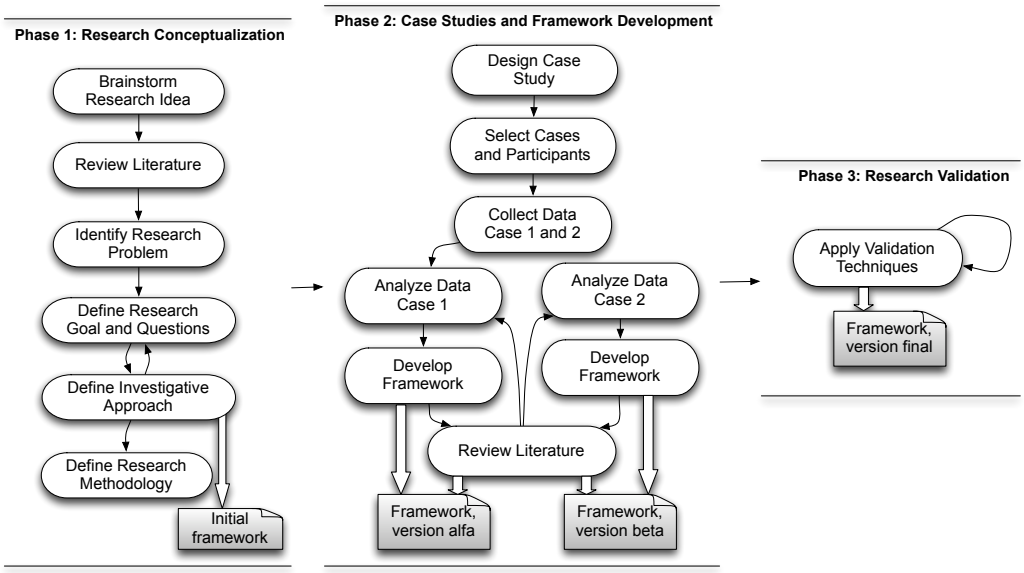


Figure 3.1: Research design

driven collaboration and on the examination of communication and fleeting knowledge in the coordination driven by requirements. I empirically investigated communication and fleeting knowledge in two industrial projects aiming to answer the posed research questions. Phase 3 comprised the research validation which was conducted throughout the research process and in parallel with Phase 2. In this chapter, I introduce the activities performed in each phase of this research. Details about data collection and analysis methods and processes are presented in Chapter 5 after the introduction of the framework for studying requirements-driven collaboration in Chapter 4.

3.1 Phase 1: Research Conceptualization

The goal of Phase 1 was to conceptualize the research. The first step involved **brainstorming ideas** within the Requirements Engineering discipline based on my supervisor's expertise in the field and motivated by problems I have experienced in industry while working with and for software development companies. I got interested in learning about collaboration driven by requirements based on the fact that software development is inherently a collaborative endeavor, and that software projects are characterized by constant changes to requirements specifications [94]. These changes need to be timely reflected in the source code if impact on quality and losses in developer productivity are to be avoided.

A brief **literature review** revealed that coordination is considered to be one of the major causes of failure in software projects [84], and most of these failures have their cause originated by requirements and software design issues [63]. Poor communication between people and lack of appropriate knowledge or shared understanding are identified as main causes of failures in requirements engineering [92]. The reason why effective communication and up-to-date fleeting knowledge has been notoriously difficult to achieve is that a software team consists of members who differ on levels of background and knowledge about the project requirements. These differences require that team members put in additional effort to complete the work. Team members have to first establish a shared understanding and a common ground about the requirements to then coordinate the work to be done.

Based on this initial literature review, which aimed at exploring the topic and at discovering relevant material, I identified that there was a lack of in-depth empirical understanding of the communication and fleeting knowledge that underlying the collaboration driven by requirements in a software project. The **research prob-**

lem to be addressed was then set. Next, I **defined that my research goal** was to learn more about communication and fleeting knowledge through their empirical examination in industrial projects aiming to address the identified problem.

A second immediate round of literature review aimed at identifying approached to investigate collaboration in software projects. The search brought out Cataldo et al.'s [24] and Erhlich and Chang' s [48] work that use social network theory to investigate collaboration behavior in software development teams. Social networks analysis examines the structure of social relationships in a group to uncover the patterning of people's interaction [99]. Inspired on the novelty of their approaches in the software engineering field and motivated by the power of social network analysis in identifying group behavior and patterns of interaction, I **defined that social networks would be an appropriate approach to examine** the communication and knowledge management processes in requirements-driven collaboration. This definition led me to extend my research goal to include now a proposal of a way to formally study collaboration driven by requirements. The **research questions were derived** from the research goal as I furthered the review of literature and consolidated a substantial knowledge in the areas of interest. These questions pertained to identifying, for example, the network structures formed during collaboration driven by requirements and which information exchange patterns are found when examining requirements-driven collaboration.

Literature review continued during the remaining time of my degree. A third focused round of review was incrementally conducted during the data analysis phase of the multiple case study aiming at identifying recent literature on the related fields of investigation to support the empirical examination of requirements-driven collaboration in the two projects and, ultimately, to further the development of the framework.

Next, I **defined the research strategy** that was appropriate to adopt in order to conduct the empirical investigation of my research. I chose case study since I wanted to explore the collaboration driven by requirements that takes place in a software project in practice. Before I discuss about case study as a research strategy, I briefly introduce the investigative approach defined to examine collaboration driven by requirements based on social network theory. The complete definition of this approach, which I call from now on *framework*, answers research question 1 that asked *"How can one investigate requirements-driven collaboration?"*. In Chapter 4, I present the initial version of the framework for studying requirements-driven collaboration. The main idea behind this initial framework was conceptualized in collaboration with

my supervisor and Irwin Kwan, a PhD student in the research lab I am associated with. The initial framework was used to guide the design of the case study, and I incrementally improved and revised it throughout my research project until it reached its final form presented in Appendix D.

3.1.1 A Framework for Studying Requirements-Driven Collaboration

Requirements-driven collaboration is collaboration that occurs during the elicitation, definition, specification, implementation, testing, and management of requirements. To study requirements-driven collaboration, I proposed a framework that uses concepts and measures from social network analysis [135] to obtain insights about the communication and coordination patterns of those involved in requirements-driven collaboration. The framework is based on a social structure that focuses on the requirement as the unit of work around which collaboration occurs. I term this structure a requirements-centric team.

A **requirements-centric team** (RCT) is a cross-functional group whose members' work activities are *related to* one or more interrelated requirements, as well as downstream artifacts such as design, code and tests. By *related to* I consider relationships such as *assigned to* and *communicating about*.

I define a requirements-centric social network to analyze the collaboration within requirements-centric teams. A **requirements-centric social network** (RCSN) is a social network [135] that represents the members, also called *actors*, and relationships, also called *ties*, in a RCT. The actors in an RCSN are among the members of the RCT, and the ties in the network are representations of different relationships during these members' collaboration. For example, a tie can represent project members' requirements-related communication, assignment to work on the same requirements, contributions to the development of a requirement, or awareness of another's requirements-related work.

Having defined requirements-centric social networks, the framework defines an initial set of **measures** from social network analysis as mechanisms to explore collaboration driven by requirements. These measures were selected based on literature review and guided by the posed research questions. Later on, more measures were added to this initial set of measures based on the empirical investigation of requirements-driven collaboration in the two investigated projects. The complete list of social network

measures that compose the framework for studying requirements-driven collaboration is presented in Chapter 8.

3.1.2 Case Study as Research Strategy

In my research, I adopted a multiple exploratory case study approach. A case study is an empirical study that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between the phenomenon and context are not clearly evident [140]. The case is bounded by time and activity, and researchers collect detailed information using a variety of data collection procedures over a sustained period of time [123]. The exploratory type of case study is used to help a researcher understand a research problem, and it is usually used where there is little in the literature about a topic [140].

Case studies differ from experiments in that the context in which the study is performed is not under the control of the researcher, i.e., the confounding factors that may impact the result are not entirely known or cannot be controlled. As not all variables in a case study can be controlled, it is difficult to claim the validity of relationships found, as they can always be claimed to depend on a possibly unknown confounding factor [17].

Yin [140] suggests two ways of coping with this problem, as follows: (1) conduct multiple case studies, or (2) use triangulation to gather cooperative evidences in a single study. "Triangulation" is a term that is borrowed from geometry and implies that multiple evidence is used to justify a new finding, just as the known location of multiple points in an n-dimensional coordinate grid can be used to determine the location of an unknown point, with increasing accuracy depending on the number of known points. The first approach is difficult to attain in practise because it is hard to get access to multiple cases in software engineering, but triangulation is a compelling approach as it compresses the lead-time needed to study a phenomenon and does not require the need for multiple cases [17].

Another issue usually associated with case studies is the common misunderstanding that one cannot generalize from a single or few cases, therefore, the single- or few-cases study cannot contribute to scientific development [56]. One can generalize from a single case study when the case is properly chosen and representative of the category [140]. Even if a single case study could not be formally generalized it does not mean that knowledge cannot enter into the collective process of knowledge accu-

mulation in a given field or in a society. A purely descriptive, phenomenological case study without any attempt to generalize can certainly be of value in this process and has often helped cut a path toward scientific innovation [56]. Flyvbjerg [56] argues that a scientific discipline without a large number of thoroughly executed case studies is a discipline without systematic production of exemplars, and a discipline without exemplars is an ineffective one. Software Engineering as a new science needs as many case studies as possible.

In my research I used both approaches suggested by Yin [140] aiming to increase the reliability and the validity of the results. Two cases were investigated, and multiple data sources were consulted through the adoption of diverse data collection methods. Details about the design of the multiple case study, the criteria for selection of the cases and participants, and data collection and analysis methods are presented next.

3.2 Phase 2: Framework Development and Case Study

The goal of Phase 2 was to further the development of the framework for studying requirements-driven collaboration and to empirically examine communication and fleeting knowledge underlaying collaboration driven by requirements in industrial projects. To achieve this purpose, given the initial framework, I moved on to conducting the multiple case study. I started by designing the case study, and by selecting the cases to be investigated and by identifying who were the target participants. Next, I conducted multiple on site visits to collect data for both projects. The collected data were incrementally analyzed by project in parallel with the incremental development of the framework.

3.2.1 Case Study Design

In this step I designed how I was going to conduct the multiple exploratory case study in order to answer the research questions. The main purpose of a research design is to help to avoid the situation in which the collected evidence does not address the initial research questions [140]. Therefore, I first identified which data was relevant and needed to be collected. Software requirements and their characteristics, communication interactions, and awareness level of requirements-related matters were listed as

necessary information to be collected. An overview of each project, organization and team structure, and organizational processes and policies were also listed as necessary. I mapped which piece of information was going to answer each research question, and then moved to the definition of which data collection methods I should use to gather the data.

Despite that it was initially less developed than the data collection strategy, the definition of a data analysis strategy was also scope of the case study design step. Yin [140] recognizes that the analysis of a case study evidence is one of the most difficult aspects of doing case studies but he claims it is necessary to avoid not achieving the research goal. At this moment, I required that data be stored in a format that would allow me to easily construct and manipulate the social networks later in the project. I also defined that UCINet 6.0 was going to be the commercial tool used to conduct the analysis of the requirements-centric social networks because this tool implements the calculation of most of the social networks measures that I had initially defined as part of the framework for studying requirements-driven collaboration. Statistical and descriptive analysis were also identified as necessary to partially answer the research questions. The selection of the cases and the participants took place next.

3.2.2 Cases and Participants Selection

To understand how team members coordinate work driven by requirements and as a consequence to answer the posed research questions, I defined a minimum set of criteria to select which cases to study. These criteria were applied to a set of projects opportunistically made available to me through industry contacts in Brazil, and are as follows:

- **Project type and life-cycle phase.** My main interest was to study new software development due to the opportunity to observe team members conceptualizing the software and defining the requirements from the very beginning. However, since the maintenance of legacy software or the enhancement of software features also require the definition of requirements, I also considered these two types of project. In addition, I defined that the project could not have completed the requirements negotiation phase. I restricted the project selection to this early phase of the development life-cycle in order to be able to observe distinct teams and roles collaborating. There was no restriction regarding the

development methodology (e.g., waterfall, iterative, spiral, agile) followed by the project team.

For the two companies that I negotiated access to conduct my research, one made available one new software development project that was further in the development phase, which I will name *candidate project 1*, and two maintenance projects that were about to start, named *candidate project 2* and *candidate project 3*. The second company offered a new software development project that was part of a largely-distributed legacy system which had just finished defining the main goal of the project and signed the development contract with the business partner. I will name this project *candidate project 4*.

- **Percentage of distribution of team members.** To study whether distance is a factor that affects coordination of work driven by requirements, I focused in selecting projects in which the physical distribution of the team members is of at least twenty percent. For instance, if a project has ten members allocated to work on it, at least two of them have to be physically dispersed from the others. In addition, I aimed for selecting projects where team members belong to different functional teams in order to investigate whether organizational distance affects coordination.

  All four candidate projects had their team members physically distributed. *Candidate project 1*, *candidate project 2*, and *candidate project 3* were mainly distributed between Brazil and the US company's headquarters office. *Candidate project 4* was distributed between Brazil, Canada, France, and Portugal. All four projects were composed of at least the following roles: requirements analysts, developers, and testers. For *candidate project 1* the development leader were going to play the requirements analyst role.

- **Easiness to reach remote team members.** Since my case study included conducting a field investigation, I needed to assure that I could reach all team members, including those located in remote sites from where I was going to conduct the study (Brazil). Therefore, I decided to select projects where it was possible to identify a priori if I could contact the remote members. It is important to note that participation in my research was on a voluntary basis, thus the fact of being able to contact a team member was no guarantee he was going to participate in the study.

For *candidate project 1*, *candidate project 2*, and *candidate project 3* the remote US members were described as accessible by their Brazilian colleagues. A relevant percentage of them had previously being involved with research conducted by research partners of the company, and were familiar with reasons why researchers ask them to participate in interviews or to answer questionnaires. Members of the *candidate project 4* located in Portugal were described as accessible by their Brazilian colleagues, however the Canadian and French members were unknown to the Brazilian team since they were contractors and new to the team. Thus, a further contact with the company focal points in both countries suggested that these members would likely not have time to participate in the research since it was known up-front that they had a short schedule to develop critical features of the new software.

- **Language.** The language selected to be spoken among and used to write by the team members was the English since this dissertation was developed in an English-speaking country. Additionally, Portuguese was defined as an acceptable spoken language. I am fully capable of understanding verbal communication in Portuguese since it is my native language.

  Although Portuguese was the native language of the members located in Brazil for the *candidate project 1*, *candidate project 2*, and *candidate project 3* projects, English was the official written language adopted in the respective company. English was also used in verbal communication with remote members for these projects. *Candidate project 4* used Portuguese from Portugal as the official language between the Brazilian and the Portuguese offices, and English between Canada and France. Three senior members in Brazil fluent in English were translating all written communication between the four sites, and a professional translator was hired to translate eventual communication that could take place between Brazil and France.

Based on the attendance of the selection criteria, two cases with distinct characteristics were opportunistically selected which are: the *candidate project 2* and the *candidate project 3*, both from the same large international manufacturing IT company named ORG from now on for confidentiality reasons. Candidate project 2, the *Shipping System* project, is a maintenance project that was about to start when I arrived at ORG. More than 20% of it members were located in a remote site, and these members were indicated as open to participate in the research. English was the

official language adopted in the project. Similarly, candidate project 3, named *Support Applications* project, was also a maintenance project which was about to start. Its members were located in three different offices in the city I was going to conduct the study, and English was the official language adopted in the project. There was no concern about the participation of the remote members since I had access to all of them.

Candidate project 1 was discarded because it did not attend the project life-cycle phase criteria. This project was further in the development phase. Although *candidate project 4* did not fully attend the language criterium, it was initially selected and data collection took place for a short period of time for this project. Since I had access to the official French-Portuguese translator, I assumed that it was acceptable to open an exception and investigate this project. In my first observation sessions I realized that it was not going to be an easy task for the project team members to overcome the language barriers, and soon I confirmed that there was too many misunderstood situations caused by language issues and that the lack of uniformity for the language adopted was going to jeopardize the data collection and analysis process. The rejection of the invitation to participate in my research from the French members and most of the Canadian members corroborated my decision of dropping this project.

Both selected projects *Shipping System* and *Support Applications* attend ORG needs to support its business processes, and the software is for internal use only. Software development at ORG takes place in the IT development centres located in Brazil and India, and in the headquarters located in the United States. ORG has over two thousands employees working in development software, and about six-hundred in the Brazilian unit. ORG develops new software products, and maintain or enhance actual applications. On average, over two hundreds projects are development annually at ORG. Most of these projects are developed in partnership with the Brazilian unit. At ORG, the projects customers are either its employees or ORGs business partners or contractors. In 2003, ORG initiated a project aiming to unify and to align the development processes of ORG's development centres. Processes and tools were standardized within a year. However, early in 2006 organizational changes in the IT department set that each business area was free to define which processes to follow. A set of tools to support software development and management was made available and each business area was granted the freedom to decide how to institutionalize their use.

Despite the fact that *Shipping System* and *Support Applications* projects are from the same company, they attended different business areas and reported to distinct senior management. These factors, in addition to the individual project characteristics, offered me the diversity I was looking for.

The totality of team members allocated to each selected project was invited to participate on a voluntary basis. The diversity of roles played by the team members and the background of each participant was relevant to compare and contrast the data collected and, consequently, to enrich the understanding of requirements-driven collaboration in practice and of the contributions of this multiple case study.

3.2.3 Data Collection and Analysis Methods

Multiple data collection methods and data sources were adopted in my multiple case study. I used document analysis to collect data necessary to develop the baseline for the referential step of my research which is to construct the requirements-centric social networks. Project requirements, team members, and task allocation information were gathered from project documentation. Questionnaire was the main source of social network data collection. By using this method I collected communication and awareness data to construct the social networks. Work diaries were also used to collect communication data. Data from this method was used to corroborate the questionnaire data only. Contextual information about the projects was collected through semi-structured interviews and observations. These contextual information helped in the interpretation of results of the social network analysis of the requirements-centric social networks.

The main data collection activity was three months long (October 2006 to January 2007). However, I visited the project teams on site three more times throughout the three remaining years of my degree (December of each year). These visits aimed at presenting partial results and discussing their usefulness, as well as brainstorming with team members which other aspects of requirements-driven collaboration they would be interested in learning from me. I also gathered additional data by e-mail whenever clarifications were required. Data collection is discussed in details in Chapter 5.

I have used three techniques to analyze the collected data. Social network analysis was used to examine the requirements-centric social networks. It was the predominant analysis method that was adopted in this research. Interview data analysis and statistical analysis were used to support the interpretation of the social network anal-

ysis results. Statistics was also used to analyze some non-social network questions from the questionnaire. These questions were used to complement the description of the contributions of this dissertation. The data analysis followed an incremental approach where more social network analysis measures were applied as preliminary results were found. The data analysis is discussed in detail in Chapter 5.

3.3 Phase 3: Research Validation

I relied on three criteria proposed by Strauss and Corbin [126] to validate the quality (innovation, usefulness) and the credibility (trustworthy, reflection of the phenomenon characteristics) of the contributions of my research: *fit*, whether results resonate with the audience (researchers and/or practitioners) for whom the research was intended to; *applicability or usefulness*, whether findings offer new insights about the investigated phenomenon; and *sensitivity*, whether data and findings were derived from research questions or research questions were posed inspired on the analysis of data collected. Throughout the research process I adopted the set of validation techniques suggested by Creswell [32] to validate the three criteria. These techniques are discussed in detail in Chapter 9.

Chapter 4

A Framework for Studying Requirements-Driven Collaboration

Requirements-driven collaboration is collaboration that occurs during the elicitation, definition, specification, implementation, testing, and management of requirements. To study requirements-driven collaboration, I propose a framework that uses concepts and measures from social network analysis [135] to obtain insights about the communication and awareness patterns of those involved in requirements-driven collaboration. The framework defines an investigative approach based on a social structure that focuses on the requirement as the unit of work around which collaboration occurs. I term this structure a requirements-centric team. The framework then consists of two parts:

- Part 1 defines the requirements-centric team and requirements-centric social network concepts, and
- Part 2 defines a number of social networks analysis measures to study aspects of requirements-driven collaboration.

4.1 Part 1. Defining the Concepts

4.1.1 Defining Requirements-Centric Teams

A **requirements-centric team** (RCT) is a cross-functional group whose members' work activities are related to one or more interrelated requirements, as well as downstream artifacts such as database design, source code and test cases. By *related to* I consider relationships such as *assigned to* and *communicating about*.

The membership of an RCT contains individuals that have a relationship to a requirement or multiple interrelated requirements. Such relationships also include relationships to downstream artifacts that trace to the requirement. Thus, the RCT membership includes individuals who work on project artifacts such as database design, source code and test cases, as well as individuals who send and receive communication artifacts such as e-mail and instant messages. As an example, consider a project team comprised of team members Bob, Eva, Frank, Geoff, Lisa, Ron and Todd, and a number of requirements R_1, R_2 and R_3. The following activities and relationships have been recorded: Lisa, a software designer, is writing a design specification implementing R_1, as well as test cases for R_1. Todd has written source code that implements R_1. Eva exchanged an e-mail message with Todd during their work about R_1. Consequently, the RCT associated with R_1 (R_1CT) contains Lisa, Todd, and Eva. This is illustrated in Figure D.1 that shows R_1 on the requirement plane, its associated project and communication artifacts on the artifact plane, and the R_1CT on the requirements-centric teams' plane.

A requirement-centric team can also provide a view of people who are working on multiple related requirements. If a requirement is related to another through requirement dependencies such as structural (e.g., refined-to, changes-to and similar-to dependencies), constraining (e.g., requires, and conflict-with dependencies) or cost/value (e.g., increases/decreases cost of dependencies) [38], the requirements-centric team associated to the interrelated requirements comprises all project members whose work activities relate to these requirements and their related downstream artifacts. Figure D.1 also illustrates the RCT associated to R_2 & R_3 (R_3 depends on R_2), so the $R_{2\&3}$ CT contains Eva, Todd, Ron, Bob, Geoff and Frank.

The RCT also applies to non-functional requirements. As non-functional requirements often have a relationship with functional requirements, and cross-cut many artifacts, an RCT can identify people who should collaborate because their work on

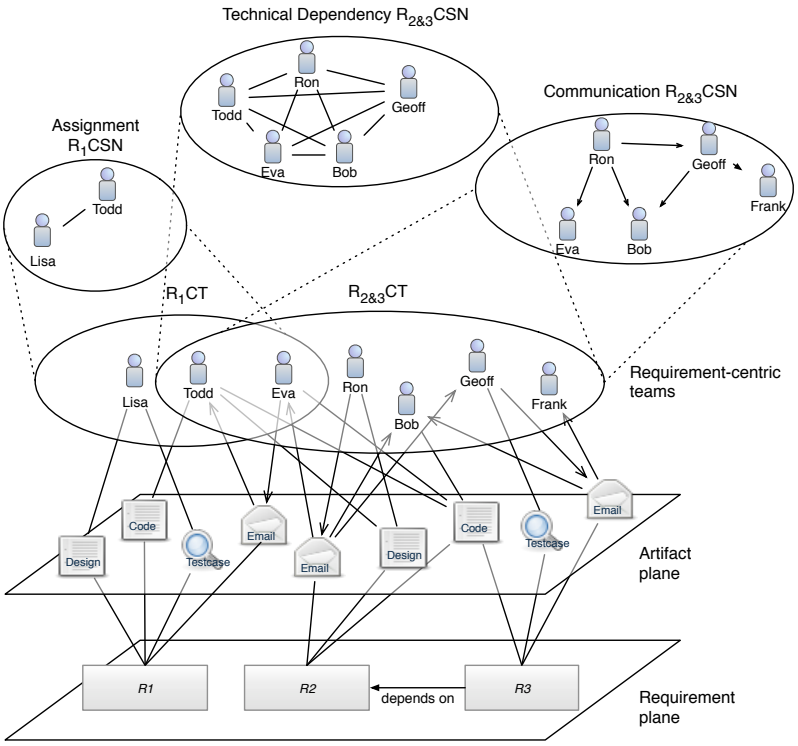


Figure 4.1: Requirements-centric teams and different RCSNs

non-functional requirements influence those working on functional requirements.

4.1.2 Defining Requirements-Centric Social Networks

To analyze the collaboration within requirements-centric teams, I define a requirements-centric social network. A **requirements-centric social network** (RCSN) is a social network [135] that represents the members, also called *actors*, and relationships, also called *ties*, in a RCT. The actors in an RCSN are among the members of the RCT, and the ties in the network are representations of different relationships during these members' collaboration. For example, a tie can represent project members' requirements-related communication, assignment to work on the same requirements, contributions to the development of a requirement, or awareness of another's requirements-related work.

Representations such as social networks allow one to capture information about the real world relationships that form among people whose work is related to a requirement, and investigate questions such as *"Who has worked on artifacts related to particular requirements?"*, *"How does this compare to the project plan?"*, *"Who*

communicated or coordinated about these artifacts?", and *"Who are central people in the requirements-based communication and thus are key people in the processes of expertise seeking?"*.

Given specific research interests, one can define which types of relationships to represent in an RCSN, and collect appropriate data with to generate RCSNs containing different relationships. Examples of RCSNs that represent relationships include but are not limited to the following (Figure D.1 is used for illustration).

Technical dependency RCSN. A technical dependency RCSN contains members that should coordinate because there are technical dependencies among the artifacts they work on, e.g., those that contribute to the requirement and related downstream artifacts up to the current moment in time. This network is assumed to be fully connected, and ties are non-directional. In Figure D.1, there is a technical dependency between R_2 and R_3. Because Eva, Todd, Ron, and Bob are assigned to work on R_2, and Geoff on R_3, the technical dependency network contains all five team members. Such a network can be constructed using repository mining that identifies relationships between artifacts, such as call graphs and trace links.

This network can be useful for identifying how many project members have been involved in modifying the requirement or associated downstream artifacts. The information captured in this network can be used to propagate change information to members working on the requirement, and, more significantly, members working on dependent requirements. If one's work is affected by a dependent requirement, one has to receive information of changes about the related artifacts. Other uses for this network include expertise seeking to find members who recently worked on an artifact related to the requirement, and monitoring the amount of activity in the development, to identify requirements that may require additional resources.

Assignment RCSN. An assignment RCSN contains members from the RCT that have been assigned to work on the requirement or on its associated downstream artifacts. The network is assumed to be fully connected because it reflects technical dependencies among members who should coordinate with each other, and ties are non-directional. For example, in Figure D.1, Lisa is assigned to work on the design for R_1, and Todd is assigned to coding the modules related to R_1. Consequently, Todd and Lisa appear in the $assignment{-}R_1$CSN. Such a network can be constructed by extracting data from project planning or bug-tracking systems that contain informa-

tion about work assignment.

This network can be useful for identifying the expected scope of involvement and coordination in the development of a requirement. When constructed over a period of time, this network can show changes on allocation of members in a certain requirement and this information can be used by senior project management to restructure functional allocation of members in a department or in the company.

Communication RCSN. A communication RCSN contains members from the RCT that have communicated about the requirements or its associated downstream artifact. A directional tie is drawn if one person communicates about the requirement with another person. To construct a communication network, data can be automatically extracted from communication repositories such as mailing lists, online forum systems, instant messenger logs, and comments on bug-tracking systems, or self-reported by team members through daily log diaries and questionnaires.

This network can be useful for identifying communication activity generated around a requirement, and an indication of behaviors such as asking for clarifications on requirements and communication of changes. This network can be quite larger than the technical-dependency or assignment-RCSN in that it may include members who emerge as relevant to the coordination driven by the particular requirement – for instance, by having provided technical expertise – but who do not belong to the technical network because they have not modified any technical artifact or to the assignment network because they were not assigned to work on the related requirements or downstream artifacts. Frank is, for example, an emergent person in the $communication{-}R_{2\&3}CSN$ in Figure D.1 because Geoff is communicating with him and he was not assigned to work on R_2 or R_3. Similarly, fewer members than those in a technical-dependency or assignment relationship may be communicating during the project, indicating a possible lack of coordination in the development of the requirement. Figure D.1 shows Bob and Eva as not having communicated in a technical dependency relationship. Because there may be different reasons for communication, such as communication of changes [39], coordinating activities [107], and requesting clarification [108], one can construct and analyze networks that capture only a particular reason for communication. These reasons can be extracted from repositories or explicitly asked to members when data is collected through self-reported methods. It is important to be precise in the relationships mapped to clearly understand communication patterns and be able to improve collaboration [34].

Awareness RCSN. An awareness RCSN contains members from the RCT that have been identified to have awareness about other members and their work in the RCT. Awareness is the knowledge that one has about others and their working activities. Examples include knowledge of what is going on in a task in areas that affect that member's work [39] [136]; knowledge of which team members are around, where and when, as relevant for the task [52]; knowledge of how other members can help one in his work [48]; or knowledge of changes made on a project documentation artifact such as requirements specification. In the RCSN a directional tie is drawn if one person has awareness about the other, using the different types of awareness. To construct an awareness-based network, data can be collected through interviews or questionnaires. A question of the form *"Are you aware of this project member's current tasks?"* or *"Are you aware of how can these project members help you in your work on requirement R?"* can be asked of individuals in a project team. Data can also be collected through observation of team members in their work environment.

This network can be useful for identifying who in the organization is knowledgeable about activities that surround one's work. Since coordination of activities is a critical component of collaboration in requirements-centric teams, and awareness plays an important role in facilitating coordination, information about the extent to which members in an RCSN have awareness of each other's work is useful in diagnosing the collaboration ability of members in RCSNs. This network can be different than the communication network because people may become aware through other means than communication. For example, members developing code related to a requirement may stay aware of progress by subscribing to the code repository notification feature. On the other hand, a project manager may stay up-to-date about what is going on in the project by reading status report of member's activities.

4.2 Part 2. Defining Social Network Analysis Measures to Study Requirements-Driven Collaboration

Having defined requirements-centric social networks, an initial number of measures and tests from social network analysis was selected as mechanisms to explore the different aspects of requirements-driven collaboration that this research is interested

on, posed by the research questions. These measures were selected based on literature review. Below I present the initial list of measures organized by requirements-driven collaboration aspect along with the respective theoretically-informed insights I could potentially obtain by the application of each measure.

Network and actor characterization. In order to characterize communication and fleeting knowledge networks that form around the development of a requirement or a set of dependent requirements I selected the following measures or social network concepts.

- **Sociogram and actor attributes.** The visualization of a sociogram that shows attributes can reveal patterns of relationship behavior among people [135], bring hidden structural features to light [80], and help on the understanding of the social processes that generated the network structure. The inspection of a sociogram with attributes can also inspire the definition of hypotheses about the social behavior of actors in a network that may be tested using social network measures [66].

- **Network size.** It may suggest the amount of collaboration required. The number of members in a requirements-centric social network can be characterized by the members' attributes. For instance, one can calculate the distribution of members by country. Moreover, if contrasted with the assignment network, one can identify which members that are part of a certain requirements-centric social network were not assigned to work on the requirements. These members are named *emergent members*.

- **Network density.** It reveals the amount of collaboration that took place among people out of the possible capacity. The result has to be carefully interpreted since not always all the possible relationships have to be fulfilled in a social network. For example, in a network representing communication of a large group a high density may be detrimental to the project performance since it will be rather difficult for the members to maintain contact with all colleagues. Also, duplications may happen if team members have multiple channels of information. This measure aims at suggesting how much collaboration took place only.

- **Ties statistics based on actors attributes.** The statistics about the relationships established by team members based on their personal characteristics may make explicit, for example, how many ties are cross-sites or cross-teams characterizing whether collaboration takes place beyond geographical and team boundaries. If contrasted with the assignment network, one can identify which ties are between two assigned members or between an assigned and an emergent member. The tie between an assigned and an emergent member is named an *emergent tie*.

Network structure. To examine which structures form during the development of requirements-related tasks, I chose the **network centralization** measure. Network centralization can reveal how tightly the network is organized around its most central points [117].

Information exchange and knowledge sharing. To study how information is exchanged and knowledge is shared among members who possess specific knowledge about requirements, I chose to use the following measures.

- **Component.** The component test reveals whether there is a potential division in a network.
- **Centrality.** Central people tend to have more influence in the network than others. *Degree* centrality suggests level of activity [59].
- **Brokerage.** A broker actor is in a position to manage or broker information flow. A broker can be problematic if intentionally or unintentionally the actor introduces misunderstandings or limits exchange of information. Because I was interested in in examining flow of communication across dependent requirements in which an overlap between requirements may exist, I customized the brokerage measure defined by Gould and Fernandez [61]. This customization allows me to account for the overlap between a pair of dependent networks. Section 4.2.1 introduces the customized brokerage measure in details.

Networks alignment. To examine the extent of the alignment between coordination needs and the teams ability to coordinate work, I adopted the **socio-technical congruence** measure proposed by Cataldo and colleagues [24]. The formal definition of this measure has been presented in Chapter 2.

In order to examine the extent to which requirements-driven collaboration patterns are aligned with the organizational structure, I extended the current socio-technical congruence measure to take into account the imposed organization structure as defined by the communication channels allowed between pairs of roles. I name this extended measure *Role-based socio-technical congruence measure*, and I introduce it in a separate section below.

4.2.1 The Customized Brokerage Measure

To examine the flow of communication across dependent requirements I customized the brokerage measure defined by Gould and Fernandez [61] to allow the analysis of brokerage between dependent requirements networks. In other words, I used the assignment to a certain requirement as the attribute to divide the members into groups. Then, three groups were defined: the group of members who work on the requirement that is dependent on another requirement, named *dependent* requirement; the group of people who work on the requirement that the dependent requirement dependents on, which I call *dependee* requirement; and those who work on both requirements. Since people working on both requirements also belong to either the dependent or the dependee groups, it was needed to customized the actual brokerage measure as presented below.

Consider that a member s sends information to a receiver member r through a broker member b, and the three requirements group mentioned above. The possible broker configurations defined are presented in Figure 4.2. These broker configurations characterize three information flows, named outgoing flow, incoming flow, and consulting flow.

Outgoing flow. This flow investigates the presence of brokers who mediate information from the dependee to the dependent requirement. I define that the brokered information flow with respect to (s,r) leaves the dependent network if s is assigned to

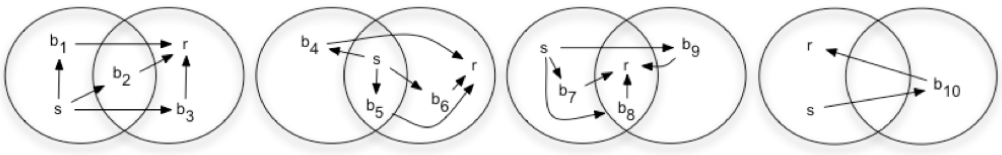


Figure 4.2: Broker configurations examined within the communication-RCSNs

the dependee requirement, r is assigned to the dependent requirement network, and a broker b for (s,r) is assigned to the dependee requirement. In short, I call this flow *outgoing information flow* with respect to (s,b,r). Outgoing flow brokers are those who fall in one of the following configuration cases in Figure 4.2: b_1, b_2, b_4, b_5, b_7, and b_8.

Incoming flow. This flow investigates the presence of brokers who mediate information from the dependent to the dependee requirement. I define a brokered information flow with respect to (s,r) enters the dependee requirement if s is assigned to the dependee requirement, r is assigned to the dependent requirement, and a broker b for (s,r) belongs to the dependent requirement as well. In short, I call this flow *incoming information flow* with respect to (s,b,r). Brokers b_2, b_3, b_5, b_6, b_8, and b_9 in Figure 4.2 relate to the configurations for this flow.

Consulting flow. This flow investigates the presence of brokers who mediate information either from the dependent or from the dependee requirement. A broker b for (s,r) indicates a consulting flow if s and r are assigned to the dependent requirement and b is assigned to the dependee requirement, or s and r are assigned to the dependee requirement and b is assigned to the dependent requirement. Brokers b_4, b_9, and b_{10} in Figure 4.2 shows possible configurations. Observe that configuration b_{10} can be mirrored.

4.2.2 The Proposed Role-Based Socio-Technical Congruence Measure

The Role-based measure considers the organizational structure defined as the set of project roles that are allowed to directly communicate with each other. Defined by the organization, a project role consists of a set of responsibilities expected to be performed by the person assigned to the role. By considering the so called organizational structure, the Role-based measure identifies congruence gap between pair of team members performing roles who, despite the coordination needs informed by the requirements dependencies, according to the imposed organization structure should not directly communicate and in fact they do not communicate. These gaps are called **false congruence gaps**, in short false gaps. For example, in a certain project the organization structure defines testers cannot directly communicate with developers.

In order to both roles to exchange information, the organization structure defines that they have to go through their leaders. Lets consider that the requirements dependencies created a coordination need between a certain tester and a certain developer. If this tester and this developer did not directly communicate with each other, then the lack of coordination between these two individuals is a false congruence gap. False gaps highlight point-to-point direct coordination needs that might not have to be satisfied because of the organization structure may compensate the need in other ways.

The Role-based measure also identifies actual communication that takes place between pair of role that should not communicate with each because of the restrictions imposed by the organization structure. Actual communication instances between people playing roles that have restricted direct access to each other are called **backchannel communication**. Backchannel communication indicates "behind-the-scenes" communication that may have taken place to compensate the coordination needs between these members. For instance, if the tester and the developer of the example above communicated with each other, the communication instance between them is considered a backchannel communication because the roles they play in the project have restricted access to each other according to the organization structure.

To identify false gaps and backchannel communication, the modification that I bring in the **Role-based socio-technical congruence measure** is the addition of a new matrix that indicates which roles are allowed to communicate with which other role. This new matrix is used in the calculations such that if no communication occurs between two roles, there is no loss of congruence.

Given k roles, we indicate which roles should be coordinating with which other roles in a $k \times k$ **role-role matrix** R by inserting a value 1 in the matrix at position ij if roles r_i and r_j should coordinate with each other. Thus, we calculate a **role-coordination matrix** OC as follows:

$$OC = OA \times R \times (OA)^t \tag{4.1}$$

where OA is an $m \times k$ **role-assignment matrix** that indicates that a person is assigned as a role, and R is a $k \times k$ role-role matrix that indicates that a person in a particular role should coordinate with another person in a particular role. The diagonal for OC is ignored.

OC is then combined with the coordination needs matrix to identify the coordi-

nation needs with roles in mind. CN is calculated as in Equation 2.1.

$$CN'_{ij} = OA_{ij}CN_{ij} \text{ for } i = 1, \ldots, m \text{ and } j = 1, \ldots, m \tag{4.2}$$

where $CN\prime_{ij}$ is the new coordination needs that incorporates the effect of roles. The value is 1 to indicate that coordination is expected, whereas a 0 means that no coordination is expected.

The actual coordination matrix is gathered from observed data the same as in Cataldo's congruence measure. However, since our new CN' may be smaller than the AC, all cases where a role is not allowed to communicate with another role must be removed before I can calculate alignment. I calculate each cell of an alignment coordination matrix AL by

$$al_{ij}(cn'_{ij}, ac_{ij}) = \begin{cases} 1, & ac_{ij} - cn\prime_{ij} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, m$. l_{ij} is 1 if there is alignment, and 0 otherwise. This equation creates a matrix where we keep connections that appear in both CN' and AC.

I use AL to calculate congruence:

$$congruence' = \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} al_{ij}cn'_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{m} cn'_{ij}} \tag{4.4}$$

The diagonal for the AL and CN' matrices are ignored, as in Equation 2.2.

In the role-based STC measure, to identify the location of a congruence gap, named from now on **real gap**, we calculate a new **lack-of-coordination matrix** G' using CN' and AL.

$$rg_{ij}(cn'_{ij}, al_{ij}) = \begin{cases} 1, & cn'_{ij} - al_{ij} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, m$, where rg_{ij} is the real gap value. A value of 1 indicates that there is a real gap, and a value of 0 indicates that there is no real gap. As with the other calculations, the diagonal is ignored. The location of a real gap allows us to identify which pair of team members did not coordinate and their

respective personal attributes.

The number of misleading gaps is calculated in the organization using the following technique. First, the lack-of-coordination matrix L is calculated as above using Cataldo's CN matrix. Then, the OC matrix is calculated with the defined technique. Finally, for each position in the matrix the following is calculated

$$mlgap_{ij} = L_{ij} - OC_{ij} \text{ for } i = 1, \ldots, m \text{ and } j = 1, \ldots, m \tag{4.6}$$

where $mlgap_{ij} = 1$ is a misleading gap, or 0 otherwise. The resulting matrix indicates between which pairs there is a technical dependency between individuals who are not supposed to talk to each other according to the OC matrix.

By incorporating the imposed organization structure in the socio-technical congruence measure, the role-based measure also allows us to identify whether backchannel communication occurs in a project. Similarly, backchannel links are identified by calculating

$$bc_{ij} = L_{ij} - L'_{ij} \text{ for } i = 1, \ldots, m \text{ and } j = 1, \ldots, m \tag{4.7}$$

where $bc_{ij} = 1$ is an instance of backchannel communication, or 0 if there is no communication. The resulting matrix indicates between which pairs there is a technical dependency between individuals who are supposed to talk to each other.

More social network analysis measures were incrementally selected and added to this initial framework as the empirical investigation of requirements-driven collaboration in the multiple exploratory case study progressed. The selection of these additional measures was based on a thorough decision process. The need to identify which measure would answer more finer-grain questions within the posed research questions raised as a result of the incremental data analysis and the findings review process with the team members. The final list of measures that compose the framework is presented in Chapter 8. Next I present the case study data collection and analysis methods and processes.

Chapter 5

Case Study Data Collection and Analysis

In order to empirically examine requirements-driven collaboration, I conducted a multiple exploratory case study of two industrial projects as previously indicated. Communication- and awareness-requirements-centric social networks were constructed aiming to identify patterns of requirements-driven collaboration. In this chapter I present in details the data collection and analysis processes and methods used to construct and to examine these networks. In addition, I describe which data was collected for each project and how the requirements-centric social networks were constructed and analyzed.

5.1 Data Collection

5.1.1 Data Collection Process

The three initial and main months of data collection (October 2006 to January 2007) started with a short period of familiarization from both sides, mine as the researcher and the team members as the research participants. A senior manager introduced me to the team leaders, and they introduced me to the remaining project members in a weekly project meeting. After spending a day meeting the participants and inviting them to voluntarily participate on my research, I started the first round of data collection. This first phase aimed at gathering information about the projects. I conducted short *interviews* with development and test leaders as well as with project managers to learn about roles involved in the project and their responsibilities; formal

organizational structure; adopted processes, practices, and tools; project terminology, scope, and history of previous releases. I also asked about remote team members and who they were, and about documentation used to support the development process. The focus of these first interviews was to acquire a broad understanding about the projects, and to gather enough information to help me define the next data collection steps.

Next, I moved to develop an understanding about collaboration practices adopted by the team members (round 2). Individual *observation sessions* were scheduled according to the members' availability. I observed each of the members working in their cubicles before moving to *observe their group dynamics* during formal and periodic project meetings. During these observations, team members referred to requirements that I was still not familiar with. Then, I asked the project managers and leaders to provide me with a copy of any requirements documentation used in the project. I also followed some team members to their cubicles after a meeting in order to follow-up on how they would implement and coordinate urgent tasks that impacted others' work, or how they would propagate information updated during the meetings to their teammates. This second data collection round lasted about three weeks.

After acquiring an understanding about the project, I moved on to further my knowledge about the project requirements and their dependencies, and to learn about who was working on each requirement. This was an intense data collection period since a proper understanding about the requirements and of the task allocation scheme was essential to construct the assignment requirements-centric social networks. These networks were used as a baseline for the customization of the questionnaire that was the main source of communication and awareness data collection. Note that in this empirical investigation I limited the examination of awareness to the knowledge that one has of what others are doing that is related to one's work. This type of awareness is defined by Ehrlich and Chang [48] as *current awareness*. I *inspected project documentation* for about two weeks until I have finished identifying who was allocated to which requirement, and which were the requirements dependencies. Interestingly, I identified some discrepancies between development and testing documentation. Thus, I conducted *interviews* and led *group discussions* to reach a consensus about my model. Meanwhile, I continued to *observe* team members discussing and coordinating requirements changes, collaborating with remote members, and reporting progress to senior management.

The third round of data collection aimed at collecting communication data. First, I deployed the *work diaries* and asked the members to fill them out daily. At first everyone was excited about recording communication data on the work diaries, but after a while people either got tired of repeating this task everyday or got too busy with the project tasks that then they started to inform incomplete data. Due to the completeness issue, I strategically decided to use work diaries as a corroboration source of information only. I later compared work diaries data against the questionnaire data, mainly to identify discrepancies regarding with whom the team members interacted and what was the interaction about. Meantime, I customized the questionnaires. For each team member, I constructed the network of people one was working with and inserted this list of people in each social network question. Observations and interviews continued in a more sparse form during the three weeks that I had spent managing the work diaries and customizing the questionnaires.

The second part of the third data collection round was characterized by the deployment of the *questionnaire*. I followed-up on the questionnaires for two weeks until I got most of the responses. The missing responses from two participants of the APP project were replaced using the available-case analysis technique [125] as presented in the following section. For two weeks short interviews were conducted to clarify some answers provided on the questionnaire, and few more observation sessions were scheduled to gather data about patterns of requirements-driven collaboration towards the end of the development life-cycle.

I was careful to schedule short interview sessions since most of the team members were interviewed more than one time. In addition to the care with the time spent with each participant, I also tried to balance the time spent observing the team members. I was concerned about the possibility of them getting used to my presence and driven their responses to what I already knew about the project. This concern was exacerbated by the fact that some of the participants were already familiar with me. I have worked at ORG in the past, although I have not directly worked with any of the research participants. My extreme care aimed at minimizing jeopardizing the data collection process for staying too long, and avoiding bias and making assumptions based on my previous knowledge of the company.

I continued data collection in a more sparse manner after this initial period of three months on site. I annually visited the team members for a few days in order to present partial results and findings, and to collect additional contextual information when necessary. These visits aimed mainly to fill in gaps on the understanding of the project

technical details, and their impact on the team collaboration behavior. I also used these meetings to corroborate interpretations of data previously collected, indirectly validating data collected through interviews and observations. Clarifications through e-mail were always provided when requested.

5.1.2 Data Collection Methods

In case study methodology triangulation is a recommended technique for increasing validity. This triangulation can be achieved by using multiple methods to assess the same information in different ways in the context of a case study, or by adopting multiple data sources [17]. The multiple data collection methods and data sources mentioned above are presented below.

Document Analysis. I mainly adopted the data analysis method to collect data necessary to develop the referential step of my case study that was to construct the requirements-centric social networks. In case studies documents are mostly used to corroborate or question data obtained from other data generation methods [102] [140]. I needed information about the project requirements and task allocation. Therefore, I gathered every requirements documentation available for each project in order to name the project requirements, their dependencies, and requested changes to the requirements. I also gathered every project planning documentation available in order to identify which team member was allocated to work in each requirement-related task.

In addition, I gathered traceability documents in order to establish relationships between requirements, technical specifications, source code, and test cases designs. This traceability information was necessary to understand the task allocation plan for each project. Moreover, descriptive documents about the product that each project was responsible for were collected to support the development of an initial knowledge about the software and the team itself as suggested by Singer [120].

Questionnaire. In my research, questionnaire was the main source of social network data collection. Questionnaires are self-reported measuring instruments to assess people's abilities, views, opinions and attitudes [105]. These instruments consist of a set of questions administered in a written format. This method is one of the most commonly used to collect social network data in addition to self-recording diaries

[29], also known as work diaries. Despite the fact that questionnaires are occasionally viewed as unreliable to collect this kind of data since they are based on the memory and perceptions of the participants [29], due to the large amount of data to be collected from a large number of people I judged that this was the most feasible way of obtaining social network data about communication and knowledge management patterns since my request to access ORG software repositories was denied. Work diaries were also adopted (presented next).

Social network questions were designed in two formats: one where I provided the list of names to the respondent and another where I told the respondent how many actors he should nominate to answer the question, respectively called *roster format* and *fixed choice* [135]. The list of names provided in the roster format questions was customized by respondent. Each respondent's list consists of a subset of the project team members, and was created based on document inspection. Only members that were assigned to work on the same requirements that a certain respondent was assigned to are included in this certain respondent's list.

To obtain details about communication interactions, I asked the team members to indicate the reason of communication in which he engaged with each other team member he selects from the provided list or the additional names he provided. I provided a list of ten reasons of communication, that are: requirements negotiation, requirements clarification, communication of changes, coordination of activities, planning, project progress, risk assessment, synchronization of code, implementation issues, and peer review. An additional blank line was provided asking the respondent to specify any additional reason.

In addition to the traditional design concerns of a questionnaire such as wording of the questions, layout of the forms, and ordering of the questions [120], the rate of non-responses was also taken into consideration. Social network analysis is especially sensitive to missing data since models of network structures assume that complete information is available on the relations to be described [20]. Non-response is even more problematic in social network analysis when the non-respondents possess characteristics that are different from those of the respondents [86]. Each missing answer introduces an additional gap in the social network under study. As a missing answer does not necessarily imply the absence of a social tie (because people may refuse to answer a particular question), the rate of non-responses should be lowered or even eliminated. To address this issue, I designed questions with similar structure and layout. This strategy aimed at reducing the respondent's cognitive effort and the

time needed to answer the questions. Refer to Section 5.2.1 for the description of how I handled missing social network data.

In addition to the social network data, the questionnaire also consisted of multiple options, scale and open-ended questions about demographics, communication media, awareness of others and related work, and changes to requirements and their effect on the respondent's work. These questions are blended with the social network questions throughout the four sections in which the questionnaire is organized in order to follow a logical order of the topics for the respondents. Refer to Appendix A for a sample of the questionnaire.

Before deploying the questionnaire, its content was validated by an expert on the requirements engineering field (a professor from the partner university where ORG is located) and by an expert on questionnaire design (another professor from the partner university). Questions were reviewed aiming to make sure that the questionnaire would indeed generate data about the concepts related to collaboration and to identify where they were a well-balanced sample of the topics I was interested in [102][78]. Duplicate and unclear questions were highlighted and later improved. Generic terms were replace by ORG naming conventions when possible, and a vocabulary section was included in the cover page of the questionnaire upon the reviewers request. The questionnaire was also pre-tested by an invited ORG requirements analyst and a project manager in order to identify whether the respondents could follow the instructions for how to answer each type of question, whether they had difficulties in answering certain questions, and how long they took to complete the questionnaire [102]. Some questions were removed since the ORG representatives indicated that the questionnaire was too long.

Work diaries. I asked all team members of both projects to complete a requirements-related communication diary that detailed who they talked to and the purpose of each interaction they had during the day. This data collection method is called work diary, and it requires respondents to record various events that occur during the day. It may involve filling out a form at the end of the day or recording specific activities as they occur [120]. Initially a template to be filled out in a computer was made available but a paper version was adopted later on as per the participants' request. It was easier for them to remember logging the interactions if the paper was at hand than locating the file in the computer.

Work diaries can potentially provide better self-reports of events because they

record activities on an ongoing basis when the event is still fresh in the individuals mind [29] rather than in retrospect as in interviews and questionnaires [120]. It has long been used in other fields such as psychology, and most recently in the study of technology use [62] and task interruptions in computing work [37]. In Software Engineering it may be challenging to get developers to fill out work diaries in a regular basis due to their usually intense and busy work days. Work diaries may interfere with respondents' work routine or performance [76]. Consequently, work diaries reports may be inaccurate. Due to the potential accuracy issues, I chose to use work diaries as a secondary social network data collection method in my research.

Semi-structured interviews. I conducted semi-structured interviews with the main intent of understanding the participants' communication behavior and level of fleeting knowledge and of identifying collaboration issues faced by the project teams, as well as to update and consolidate previous information I had about the organization structure and policies, development processes, and state-of-the-art of the tools usage. Since software documentation can become obsolete due to the so common changes to the project requirements, semi-structured interviews were also conducted aiming to make sure that the data identified through document analysis was up-to-date. When necessary, after a set of observations sessions I listed situations that needed to be clarified, and interviewed key participants of the observation sessions. These clarifying interviews were important to support the understanding of contextual information.

Semi-structured interview is the kind of interview where a pre-defined list of themes or questions to be covered guide the conversation. The flow of conversation in a semi-structured interview can change depending on the answers given by the respondent [102]. The flow of conversation in a semi-structured interview can change depending on the answers given by the respondent [102].

For each set of semi-structured interviews (in short, *interviews* from now on) about the same topic, a guide was designed and pre-tested with members of other projects under development in the organization. These members were invited to collaborate with the research due to the similarity of their roles to those played by the research participants, and to their experience with academic research. The interview guides were used in both projects to guarantee that the same desirable topics were covered.

All SHIP team members were interviewed at least one time. For APP, the project managers, team leaders and seniors developers were interviewed at least once. Additionally, a sample of developers and testers chosen based on project contextual

information were interviewed once each one. Interviews were conducted at a time convenient to the participants and in the organization building. I conducted the totality of individual interviews in the participants office or cubicle so the participant could show me documents, project-related e-mails, tools, or any other relevant item. The few group interviews conducted took place in the coffee area or in the smoking room to avoid disturbing the work environment. The group meetings aimed at clarifying discrepancies about requirements dependencies or technical information about the project. Key team members were put face to face to discuss with me the discrepancies and argue about their different interpretation over the same piece of data. The group interviews were crucial to make sure of the accuracy of the social networks and the requirements dependencies. All interviews were voice-recorded with the participants' consent and the organization authorization.

Observation. I observed team members performing daily activities in their workplace environment mainly to document how they collaborate to get requirements-related work done in practice, and how they relate their tasks and interactions back to requirements. I also adopted observation as an additional source to understand contextual information as suggested by Yin [140]. Observation consists of gathering impressions of the surrounding world. For instance, by watching what others are doing. Since what people say they do can be different from what people do [2], researchers usually use observation as a data generation method to find out what people actually do [102].

Interactions among team members in individual situations or in group meetings were observed. When observing one team member only, I stood by his cubicle wall or far behind his desk and took notes about what he was doing, with whom he was talking to and what was the conversation about. I also recorded which tools he was using and tried to capture the context in which the work or interaction was taking place. For group meetings, I sat in a corner of the meeting room to not disturb the group dynamics and took note of the team members interactions. More specifically, I recorded the order team members spoke, a summary of what they were saying, and described the interruptions made. Some group meetings took place over the phone. In this situation, a team member provided me a guest password for the conference bridge and I connected to the call as an invited participant. This team member introduced me to the meeting participants before the meeting started to ensure that the participants were aware of my presence.

Information collected during observation sessions was recorded in a paper notebook without distracting the participants. In addition, with the team members permission, I voice recorded all group meetings and conference calls when speakers were available. Since in observation observers only follow the flow of events [2], I could only capture what occurred and not the reason why things occurred as they did. To clarify some events, I asked the participants to answer some questions. In the first observations, I would ask these questions in the end of the session. But later, to attend the participants requests, I established a protocol where I would consolidate my questions and schedule a short semi-structured interview in the end of each working day. This protocol established a valuable loop between interviews and observation sessions, where time was assured to explore insights gained during the observations through interviews.

5.1.3 Data Collected

By using the data collection methods previously described I collected the following data. Details about the project setting will be presented in the respective case studies chapters. Here I provide enough information for comprehension of the data collected.

Requirements and their dependencies. From the inspection of requirements documents, I identified 20 software requirements for APP and 18 requirements for SHIP. By examining requirements-traceability documents I identified 9 sets of dependent requirements for APP and 8 sets for SHIP. The number of requirements per set varies from 2 to 8. From literature review, the dependency relationships between requirements can be of structural nature (e.g., refined-to, changes-to, and similar-to dependencies), constraining nature (e.g., requires, and conflict-with dependencies) or cost/value nature (e.g., increases/decreases cost of dependencies) [38]. I later interviewed the APP's requirements analysis and the SHIP's development leaders to validated the sets of dependencies identified. During these interviews I learned that the dependencies that fall in the "refined-to", "requires", and "conflicts-with" categories have a more critical impact on the projects, and where the dependencies that in fact were discussed and managed by the team members throughout the project life-cycle. For example, in the SHIP project one set of dependency involved the two following requirements (fictitious names are used to protect confidentiality of informa-

tion): requirement 1 defined that the description of the products shipped to a certain customer should be presented in two additional languages than English, and requirement 2 defined that the carrier company delivering the products to ORG customers should provide to the customer a delivery receipt addressing these two additional languages. This example was categorized as a "requires" type of dependency by the SHIP development leaders. Thus, I limited the study to this scope of dependencies. This limitation downsized to 4 the set of dependencies for APP and to 5 sets for SHIP. Each set of dependency has two requirements. I name the requirement that is dependent on another requirement *dependent* and the requirement that the dependent requirement dependents on I call *dependee* requirement. Due to confidentiality reasons I will refer to these dependencies as D_1, D_2, and so on.

Team members. At ORG, a certain team member can be allocated to work on a project release or on overall activities that cross all the releases, such as maintaining the development environment in synchronization with the production environment. I was interested in identifying people who were formally assigned to work in any task related to each requirement listed on the sets of dependencies. I inspected project planning documentation to identify the list of team members associated to each requirement, which consist of the requirements-centric teams.

For the requirements that are scope of this study, a total of 10 team members (out of 45) were listed for APP, distributed as follows: 2 requirements analysts (out of 4), 1 test leader (out of 1), 4 testers (out of 7), 2 development leaders (out of 5), and 2 developers (out of 20). In average, there are 4.75 team members assigned to each set of dependencies. For SHIP, a total of 11 team members (out of 14) were listed. These members are distributed as follows: 1 system architect (out of 1), 2 development leaders (out of 2), 4 developers (out of 6), 1 test leader (out of 1), and 3 testers (out of 3). In average, there are 8 team members assigned to each set of dependencies. Note that for both projects business partners and project managers were not considered since they are not directly allocated to work on the project. This decision is aligned with the definition of the assignment-RCSN, which was used as a baseline for the empirical study. However, it does not indicate that business partners and project managers do not collaborate to reach the project goal.

Organizational structure. From document inspection and interviews I found out that the organizational structure for APP and SHIP represents the division of labor by

roles, expected workflow, and chain of command for reporting progress. To increase control and productivity, both projects defined that communication should happen through the formal channels established by the organizational structure. This was strongly emphasized by the APP project managers. The organizational structure of each project is presented in their respective case study chapters presented next.

Communication interactions. No significant communication interactions were provided for 6 out of the 10 reasons I listed to the respondents, that are: planning, project progress, risk assessment, synchronization of code, implementation issues, and peer review. Later in interviews, team members explained that they presumed that the reasons *planning*, *project progress*, and *risk assessment* were associated with management tasks, that they were not in fact responsible for. Because of the misunderstanding I decided it was safer to drop the interactions for these three reasons. For the *synchronization of code* and *peer review* reasons, APP team members clarified that there was only one team member assigned to perform synchronization of code and one member to perform peer review, thus the lack of discussion about both topics. On the other hand, SHIP members explained that synchronization of code is made by two senior members, whose were the few people who reported communication about the topic, and peer review was temporarily suspended from the team schedule for the investigated release. I could not find a plausible explanation about why *implementation issues* was not discussed. Thus, I considered only the following four reasons of communication in the scope of this study: requirements negotiation, requirements clarification, communication of changes, and coordination of activities. Communication data collected for the redefined scope is entirely presented in the form of social network ties in the case studies chapters.

Contextual information. Relevant contextual data about the projects, such as whether team members located in the US have met people in Brazil, were collected and were used to support the interpretation of the results of the the social network analysis measures. These data are used throughout the case studies chapters.

5.2 Data Analysis

5.2.1 Data Analysis Methods

Data analysis aims at looking for patterns in the data and draw conclusions. There is a wide range of techniques for analyzing data. I have used the three techniques discussed below. Social network analysis was the predominant. However, without content and statistical analysis the interpretation of the social network analysis results was going to be based on speculations and lacking context.

Social network analysis. The social network perspective is the analysis method used in this research to examine patterns of interactions among team members of software projects. As previously defined, a social network is a specific set of linkages among a defined set of persons (actors), with the additional property that the characteristics of these linkages as a whole and of the actors involved may be used to interpret the social behavior of those who compose the social network [99]. As in any social study, this social behavior interpretation is usually based on contextual information about the group that one is studying [117]. To reveal collaboration patterns, social network analysis offers a variety of measures, such as network density, centrality degree, and brokerage. A complete definition of the measures used in this research is presented on Chapter 2 and their application are exemplified in the case studies chapters presented next.

To deal with missing social network data from the two respondents who did not answer to the questionnaire, I used the *available-case analysis* technique [125]. This technique includes both complete (where both respondents of a pair provided answer) and incomplete cases (where only one respondent of a pair provided answer) and uses whatever data are available for a given analysis. To use partially described links, the assumption is that if actor A describes a relationship with actor B (link $A - B$), indeed, a relationship does exist between them. Then, the link from actor B to actor A (link $B - A$) is fulfilled with the value indicated by actor A. This technique is also called *reconstruction*. The presence or strength of a relationship is simply determined by one description rather than two. This approach does not add a links to dataset where there were none. In order to adopt the reconstruction approach, two criteria should be satisfied: the respondents should be systematically similar (comparing their attributes) and the data available from respondents should be reliable descriptions of the relationships that they have with non-respondents. Reconstruction is one of the

most used approach in social network analysis.

Interview data analysis. Interviews were voice recorded and notes were taken during the interview. The interview recordings were transcribed into text documents for consultation during the interpretation phase of the social network analysis results. In my research, contextual information about the projects such what is the working history of senior members of the investigated projects or why certain requirements changes took longer to be approved than others was crucial for drawing conclusions about the results obtained by applying social network measures.

While visiting the teams on site I kept a research diary where I daily summarized the interactions with the participants and highlighted main points observed throughout the day. I constantly consulted the summarized content of the interviews and of my field notes to help me define the scope of my following interviews and observation sessions. This approach drove me to discover and develop knowledge about requirements-driven collaboration in a similar manner than adopted in ground theory [60], where the researcher builds theories based on grounded data rather than deducing testable hypotheses from existing theories [126] [25].

Statistical analysis. Although the number of respondents in each project is relatively small (8 members in APP and 11 members in SHIP), I statistically analyzed ordinal data collected in Likert-scale format through the questionnaire. Note that additional 35 participants answered the questionnaire for the APP project but their responses were not considered since they were not assigned to work on the dependent requirements that are part of the scope of the case study. I determined the proportion of responses in each category out of the total of responses I received by project. I also statistically described demographic data that support the social network analysis. For example, I described the distribution of team members by attributes such as location and role, and I plotted the distribution of communication interactions by topic of discussion in histogram format.

5.2.2 Data Analysis Process

The data analysis process was incremental. It informally started when I was still collecting data on site. Interview records and observation notes were daily reviewed to support the definition of the next data collection round. While on site I also reviewed

the questionnaire answers for completeness, and sough the participants out to clarify why some questions were returned blank. In most of the cases blank responses were present because the team member had missed an alternative.

After completing the data collection process and checking data for completeness, I defined a policy to handle with missing social network data. The available-case analysis technique was adopted and missing data constructed based on the participants' responses. Next, I entered the questionnaire data into a relational database, including the social network data, to allow automation of data manipulation and to facilitate visual displaying.

I moved to constructing the communication- and awareness-requirements-centric social networks for both projects once the relational database was created. This activity was followed by the application of a initial set of social network measures to the SHIP case. As partial and preliminary results were obtained, insights about what other social network measures could reveal led me to further my knowledge about social network analysis and to apply more measures. Most of these insights were obtained by discussing from time-to-time the results with ORG. Annually, I visited the teams and senior management to report progress and to hear from them which other aspects of requirements-driven collaboration they would be interested in learning about. Also, I periodically shared partial results with team leaders and senior developers by email to assure that I interpreted the data properly and correctly understood the context in which the identified requirements-driven collaboration patterns took place. It is important to highlight that names were disguised in any presentation of results to ORG team members in order to protect the individual's anonymity and privacy. (Refer to the Research validation section in Chapter 9 for details on these interactions with ORG members.)

This incremental social network analysis cycle was intertwined with rounds of content analysis in order to support the explanation of the results. The alpha version of the framework was developed in the end of the SHIP analysis cycle. The same procedure was repeated for the APP project, and an beta version of the framework was developed. Next, a complementary cycle of social network analysis was run for both projects. Statistical analysis of some questionnaire data was concomitantly performed and used to complement the description of the contributions of this dissertation.

5.2.3 How Data was Analyzed

More specific details about how and what requirements-centric social networks were constructed and analyzed is presented below.

Constructing the requirements-centric social networks. For each project and set of requirements dependency, I have constructed the respective assigned-, communication-, and awareness-RCSNs. Since 5 sets of requirements dependencies were investigated in the SHIP project, I constructed 5 assignment-RCSNs, 20 communication networks (5 sets times 4 reasons of communication), and 5 awareness networks. For the APP project 4 sets of dependencies were investigated, thus I constructed 4 assignment-RCSNs, 16 communication networks (4 sets times 4 reasons of communication), and 4 awareness networks. The assignment networks were used as baselines to define the requirements-centric teams for the communication and awareness networks as explained in Section 5.1.2.

Initially, the set of team members in each network was the same. However, since I asked the team members to name additional people that they have communicated with or they were aware of there are networks that have more members than those assigned to work on the dependent requirements. These additional members were named *emergent members*.

Communication and awareness data reported in the questionnaire was used to identify the social ties between pairs of members. In my research a tie is directional, or asymmetric, since being the source of the relation rather than the object is an indication of information power or reveals expertise. For instance, if team member A reported he communicated with member B about reason com_x for requirement $R_{w\&z}$ then I created a directional tie from A to B in the $R_{w\&z}com_x$ communication network. Note that even if no one reported a communication interaction with or being aware of a certain member in a certain network that certain member is still visually displayed in the network sociogram for discussion purposes. This certain member is not considered in the calculation of the social network measures since there is no tie from or to him that can be computed.

In the socio-technical congruence measure the assigned RCSNs and the dependencies between pairs of requirements were used to calculated the coordination needs matrices and the communication networks represent the actual coordination matrix. Emergent members are not considered in this measure, thus these members and their

respective ties were removed here.

Calculating social network measures for the requirements-centric social networks. Once the requirements-centric social networks were constructed I started the calculation of the social network measures. The social network analysis measures used to answer each research question are described in details in Chapter 2 and in Chapter 8. For comprehension, below I list the measures by research question. Except by the measures listed in the research question 5 (RQ5), all measures consider the emergent members in their calculations.

- (RQ2) RCSNs characterization: sociogram, size, density, and ties statistics
- (RQ3) RCSNs structures: network centralization, core-periphery, ties reciprocity, and clique
- (RQ4) RCSNs information flow: component, reachability, cutpoints, centrality, and brokerage
- (RQ5) RCSNs alignment: socio-technical congruence and Role-based socio-technical congruence

Note that the *ties statistics* measure uses statistical analysis and is not a social network analysis measure per se. The *brokerage* measure was customized from the original measure proposed by Gould and Fernandez [61] to attend the specifics needs of this research as described in Chapter 4. Similarly, the *Role-based socio-technical congruence* measure was extended from Cataldo et al. [24] measure to allow the consideration of imposed organization structures in the study of requirements-driven collaboration. The Role-based measure has been presented in Chapter 4.

Interpreting the requirements-centric social networks results. One-by-one the results yield by the application of each social network analysis measure to each of the requirements-centric social network was interpreted considering the knowledge developed during the three months on site and on additional information provided by team members and senior managers during review sessions. I particularly looked for collaboration patterns across the reasons of communication or set of dependencies. I considered the projects particularities when discussing similarities and differences on requirements-driven collaboration behavior between both projects. Next, I present the SHIP case study, and following the APP case study.

Chapter 6

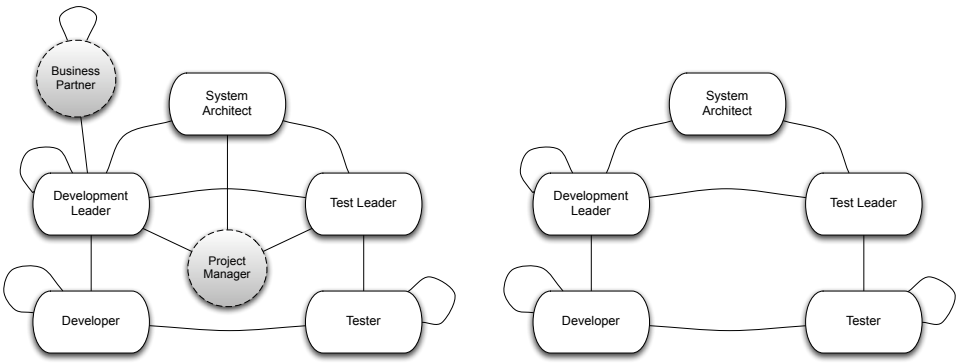
Case 1. The Shipping System Project

In this chapter I present in detail the examination of requirements-driven collaboration in the Shipping System project, contracted to SHIP. The project setting is first introduced followed by the examination of the requirements-centric social networks organized by research question.

6.1 Project Setting

Business area and project goal. SHIP, fictitious name, is a project that enhances and maintains an internal software used to support ORG's shipping process. The application is seven-years old, and is a critical component of the company's business. The project investigated was one quarterly release of the shipping application. This release aimed at keeping the product up-to-date with small changes on the business process and at improving infrastructure (e.g., database) to align it with recently adopted technology.

Project requirements. Each release enhances a set of business needs. These business needs are informed to the project team in the format of features. These features are transformed into software requirements, and recorded in a requirements specification document by the development team. The requirements are formally approved early in the project life-cycle at the end of the Planning phase, and then used to guide the remaining phases. The test team transforms these project requirements



(a) Reported organization structure (b) Reduced organization structure

Figure 6.1: SHIP's organization structure

into testing requirements for internal use only. A finer-grain level of requirements definition helps the test team to better understand the scope of its work. A traceability document aiming to keep track of the mapping among all levels of definitions (needs, features, and project requirements) is maintained in a spreadsheet format and shared with team members in the project documentation repository. The test team maintains a copy of this traceability document with the testing requirements added to it. Change requests to the requirements are usually announced by the business partners. Eventually, team members may detect an opportunity to technically improve the application. This suggestions of changes are followed-up also as change requests. After a change impact analysis, changes to the project scope are renegotiated among business partners, project manager, development leaders, and the test leader. Since this project is an ongoing enhancement to the shipping application, if the change does not affect any current functionality of the software it is scheduled to the next release in order to not affect the team's current release work.

Team distribution and office locations. The project team was distributed between the US (4 members) and Brazil (10 members) as follows: the project manager, the system architect, 1 development leader, and 1 senior developer were located in the US, and 1 development leader, 5 developers, the test leader, and 3 testers were located in Brazil. Three Brazilian developers and 2 testers were contractors and work part-time for SHIP. Contractors worked in a building five minutes walking-distance from ORG's building. Development and Test team offices were located in two different wings of the ORG's building. In addition to the project team, there were several business partners located in the US. Although the business partners were internal

clients they were not members of the project team. The project manager was allocated to work for the portfolio that SHIP belongs to but he had no specific tasks allocated to the quarterly release projects.

SHIP's organization structure, role and responsibilities. Figure 6.1(a) illustrates SHIP's organizational structure. Development leaders are in charge of gathering requirements from the business partners who work closely with the ORG shipping production team. They can also gather requirements from SHIP senior management members who work together with the ORG IT team. The ORG IT team is responsible for the IT infrastructure at the ORG plants. Development leaders also negotiate quarterly the project scope with business partners and the project manager. The project manager's responsibility is to manage the project schedule and the allocation of resources and financial budget. He liaises with the business partners if necessary. The system architect supports the development leaders regarding architecture issues when required. Since SHIP is a maintenance project there are small changes in architecture. The remaining of the team members allocated to the SHIP portfolio were working in migrating SHIP to a new technology, thus the system architect was more concentrated in the new product. The test leader is in charge of defining the test strategy for the project, and managing the test team. Note in Figure 6.1(a) that the project manager, the system architect, the development leaders and the test leader are connected to each other. Developers are responsible for coding the requirements, and testers are responsible for verifying and validating them. Observe that developers and testers have to go through their respective leader to reach the system architect, but they can directly communicate with each other when necessary.

Note that since the business partners and the project manager did not have tasks allocated to the project, I did not consider them as part of the team. For the purpose of constructing the requirements-centric social networks, I considered only team members assigned to any task related to the project requirements. Thus, Figure 6.1(b) shows the reduced organization structure considered in this research.

Team expertise and communication practices. Team members in SHIP are familiar with the product. Some members located in the US have participated on the development of the first version of the shipping system, and the Brazilian team is working together for approximately three years. The team is also very familiar with documentation and has very well-established working practices. Newcomers usually

spend sometime working in the headquarters to speed up the onboarding process and to get to know the business partners. Team members in SHIP use face-to-face interaction, instant messenger, phone, and e-mail extensively. The Brazilian development leader often meets with the test leader face-to-face, and the leaders formally meet twice a week to report status and discuss issues. Formal weekly meetings take place within and across teams. The culture of the SHIP project encourages communication among members across geographical and team boundaries.

6.2 The Examination of Requirements-Driven Collaboration

In the framework developed in this research, social network analysis measures were used as mechanisms to examine requirements-driven collaboration in the SHIP project. In practice, for each of the five sets of requirements dependencies in this project each measure was applied to each communication- and awareness-requirements-centric social network previously constructed. In this section I present the results of the application of these measures and interpret these results in terms of the project context.

6.2.1 (RQ2) RCSNs Characterization

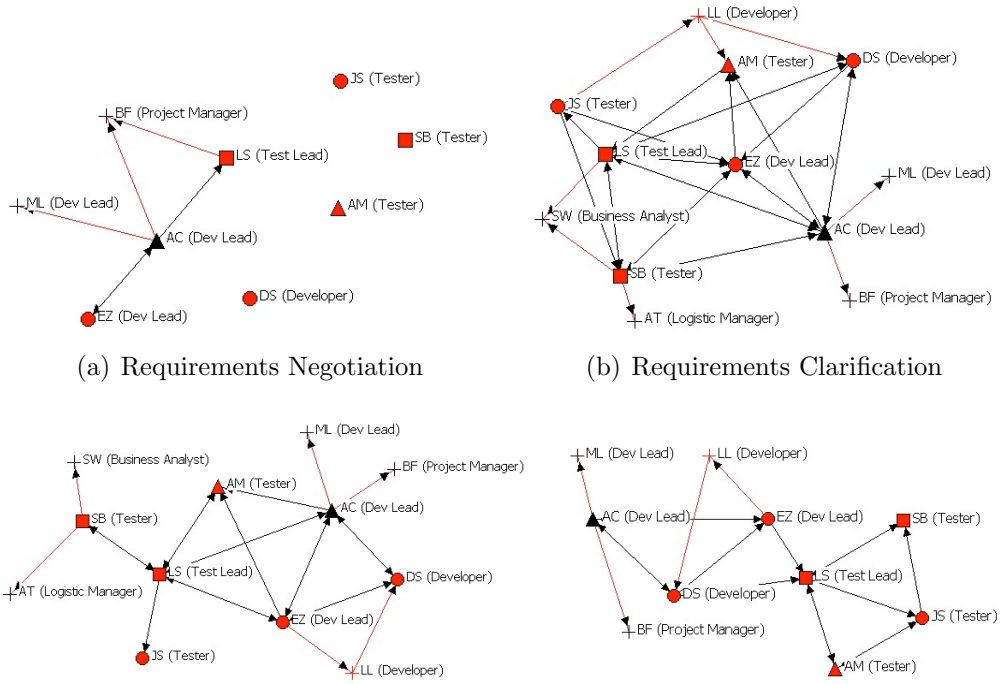
In the effort to characterize the communication- and the current awareness-RCSNs, I adopted the sociogram, network size, network density, and ties statistics measures. This set of measures was defined in the initial version of the framework for studying requirements-driven collaboration (refer to Chapter 4 for details). No measure was added to the framework to characterize the RCSNs.

RCSN sociogram. A sociogram is a picture in which actors are represented as points in two-dimensional space, and relationships among pairs of actors are represented by lines (or ties) linking the corresponding points. In addition to the actors and their relationships, sociograms can also indicate actors attributes or relationships characteristics. In the communication and awareness requirements-centric social networks constructed to examine requirements-driven collaboration in my research, the following attributes and characteristics are represented in the sociograms. The member's fictitious name and role are indicated in the actor label, for example in Figure 6.2(a) the label *DS (Developer)* indicates a person named DS who is a developer in

the SHIP project. The member's office location is indicated by the actor color (red for Brazil and black for the US), for example the member *AC (Dev Lead)* is located in the US. The requirement (or set of requirements) that the member was assigned to work on is represented by the actor shape (square for the dependent requirement, triangle for the dependee, circle for both requirements, and plus sign for emergent members who were not assigned to any requirement), for example *EZ (Dev Lead)* was assigned to work on both requirements in the dependency set and *BF (Project Manager)* is an emergent member. Whether the relationship is between members assigned to work on the requirements (black color) or with an emergent member (red color) is indicated by the color of the tie, for example the tie from *EZ (Dev Lead)* to *LL (Developer)* represents an emergent communication.

As an illustration, Figure 6.2 displays the sociagram of each of the four reasons of communication for the dependency set D_1. One can see that the sociograms are quite different from each other. For instance, although the assigned requirements-centric team for each network is the same (7 members), each network contains different members because of the emergent people who contributed to work or discuss about the requirements. The amount of communication is also distinct for each network. It varies from more dense discussions about requirements clarifications (Figure 6.2(b)) to more sparse communications about requirements negotiations (Figure 6.2(a)).

One can also see in Figure 6.2(a) that there are isolated members, for example *AM (Tester)*. These isolated people are members who were assigned to work on the set of dependent requirements and that did not communicate with or receive any communication from any colleague. It is surprising that a member who was assigned to work on a certain requirement have not communicated about it. However, this isolation is likely related to the topic of discussion. Note that testers and a developer are isolated in the requirements negotiation network because negotiation of requirements is not in their scope of responsibilities. The project development leaders, the test leader, and the project manager were the ones communicating about requirements negotiation with the business partners, which in fact is confirmed by the visual inspection of the sociogram for this dependency set. Moreover, one can also see that in the Communication of changes and in the Coordination of activities networks (Figure 6.2(c) and Figure 6.2(d), respectively) the developers and the testers seem to be clustered together within their teammates (which is later confirmed by the clique measure). In contrast to this grouping behavior, in the Requirements clarification network (Figure 6.2(b)) one can see that developers and testers have communicated with each



(a) Requirements Negotiation

(b) Requirements Clarification

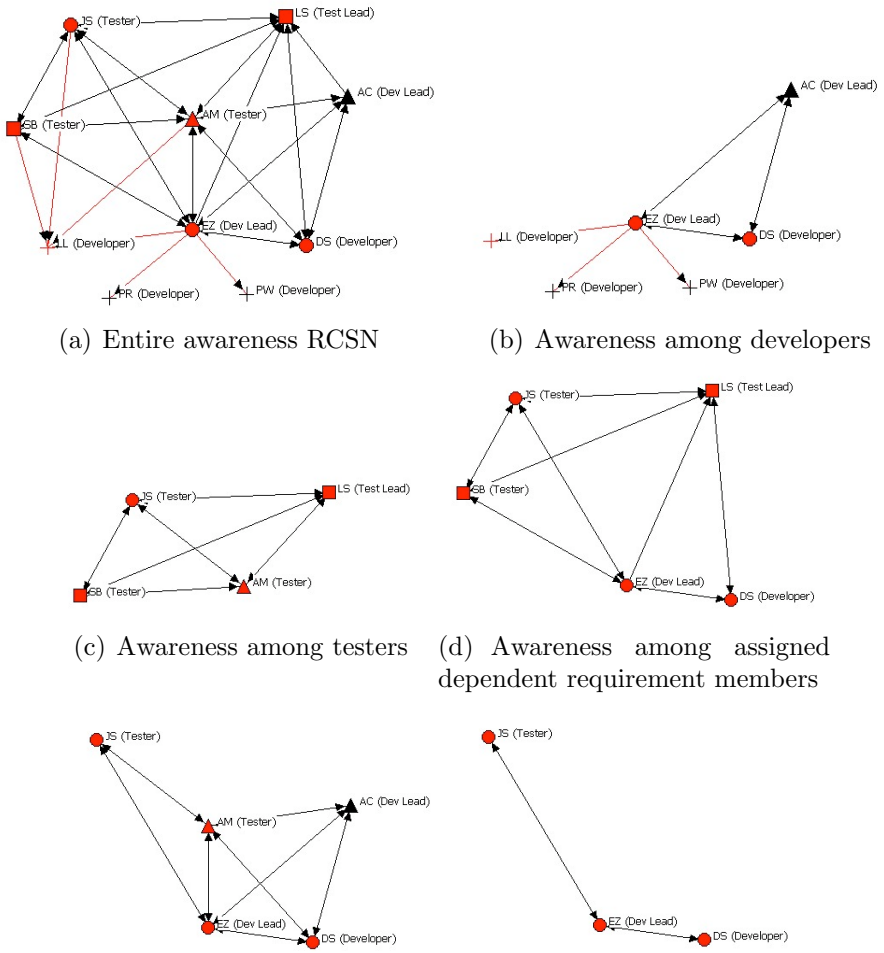
(c) Communication of Changes

(d) Coordination of Activities

Figure 6.2: SHIP's communication-RCSN sociograms

other, suggesting that members go beyond their team boundaries to clarify questions about the requirements they are working on (within and cross group communication is reported in the ties statistics analysis).

Figure 6.3(a) shows the sociogram for the current awareness-RCSN of the same dependency set, D_1. Note that this is a denser network than any of the the communication networks for this same requirements set (the density measure confirms this result later on). Distance does not appear to have impacted the ability of members to be aware of remote colleagues. Note that the only member located in the US, *AC (Dev Lead)* is aware of some of her colleagues and vice-versa. Also, developers and testers seem to be aware of their teammates. The sociograms were used to initially explore patterns of behavior such as this one. By using the members attributes I could segment the network according to the team membership (or any other attribute) to observe the behavior that I wanted to investigate. For example, Figure 6.3(b) shows which developer is aware of other developers, and Figure 6.3(c) shows the awareness relationships for the test team. Observe that both subsets of the awareness-RCSN are connected indicating that developers and testers are to some extent aware of their teammates. Similarly, Figure 6.3(d), Figure 6.3(e), and Figure 6.3(f) present the



(a) Entire awareness RCSN

(b) Awareness among developers

(c) Awareness among testers

(d) Awareness among assigned dependent requirement members

(e) Awareness among assigned dependee requirement members

(f) Awareness among assigned to both requirements members

Figure 6.3: SHIP's current awareness-RCSN sociograms

awareness relationships among members assigned to the dependent, to the dependee, and to both requirements.

Since the sociograms were used to develop an initial broad understanding of each network behavior and to explore patterns of requirements-driven collaboration later examined in details using the social network analysis measures, the remaining sociograms in the data set are not displayed in the body of this document. Refer to Appendix B for the presentation of each of the communication- and awareness-RCSN sociograms.

RCSN size. Each RCSN built for this project was initially based on a baseline network called assignment-RCSN, as defined in the framework (see Chapter 4). Each

of these assignment network consists of a requirements-centric team which is composed of a subset of the 14 team members who were assigned to the project. I refer to these members as *assigned members* from now on. To construct the corresponding communication-RCSN, I identified from the questionnaire data which listed project members from the assignment-RCSN the respondents identified as communicating about the requirement. I also identified any other emergent member that the respondents mentioned he communicated with. Therefore, each RCSN has a certain number of members, denoted *actual members*, that are divided into assigned and emergent members.

Table 6.1 shows the distribution of assigned members by location for any of the reasons of communication per dependency set, and Table 6.2(a) shows the distribution of actual and emergent members for the corresponding Requirements negotiation networks. Actual members are variable because the number of emergent members varies per network. A total of 11 emergent members across the 5 dependency sets were identified, all of them from the US. These are non-unique people, i.e. there are duplicates across RCSNs. On average, there are a total of 2.2 emergent people per network. This indicates that, on average, about one-fourth of the requirements-centric team members are emergent in a RCSN that communicated about requirements negotiation.

Table 6.1: SHIP's distribution of assigned members per communication-RCSNs by location

	Assigned	
	BR	US
D_1	6	1
D_2	8	2
D_3	7	2
D_4	7	2
D_5	5	0
$Total$	33	7
$Mean$	6.6	1.4

Similarly, Table 6.2(b) shows the distribution of actual and emergent members for the corresponding Requirements clarification networks. A total of 23 non-unique emergent members was found. On average, there are a total of 4.6 emergent people per network. This indicates that, on average, about 40% of the members clarifying requirements are emergent to the requirements dependency set. The distribution of

Table 6.2: SHIP's distribution of members per communication-RCSNs by location

(a) Requirements negotiation

	Actual		Emergent	
	BR	US	BR	US
D_1	6	3	0	2
D_2	8	4	0	2
D_3	7	4	0	2
D_4	7	4	0	2
D_5	5	3	0	3
$Total$	33	18	0	11
$Mean$	6.6	3.6	0	2.2

(b) Requirements clarification

	Actual		Emergent	
	BR	US	BR	US
D_1	7	5	1	4
D_2	8	6	0	4
D_3	7	6	0	4
D_4	8	6	1	4
D_5	5	5	0	5
$Total$	35	28	2	21
$Mean$	7	5.6	0.4	4.2

(c) Communication of changes

	Actual		Emergent	
	BR	US	BR	US
D_1	7	5	1	4
D_2	9	7	1	5
D_3	8	7	1	5
D_4	9	7	2	5
D_5	5	5	0	5
$Total$	38	31	5	24
$Mean$	7.6	6.2	1	4.8

(d) Coordination of activities

	Actual		Emergent	
	BR	US	BR	US
D_1	7	3	1	2
D_2	10	5	2	3
D_3	9	5	2	3
D_4	9	5	2	3
D_5	6	3	1	3
$Total$	41	21	8	14
$Mean$	8.2	4.2	1.6	2.8

actual and emergent members for the Communication of changes networks is presented in Table 6.2(c). There are 29 non-unique emergent members across these networks. On average, there are a total of 5.8 emergent people per network. Like in the requirements clarification networks, here the emergent members represent over 40% of all members who discussed changes in the project. Table 6.2(d) presents the distribution for the Coordination of activities networks. There are 22 non-unique emergent members across these networks. Each network has, on average, 4.4 emergent members. Likewise, these emergent members represent about 40% of those involved with coordinating activities. For the four reasons of communication, US emergent members are majority.

The distribution of team members per awareness-RCSN per set of dependency by location is showed in Table 6.3. A total of 32 non-unique emergent members across the 5 dependency sets were identified, 60% of them (19 members) from the US. These are non-unique people, i.e. there are duplicates across RCSNs. On average, there are a total of 6.4 emergent people per network. This reveals that, on average, 80%

Table 6.3: SHIP's distribution of members per awareness-RCSNs by location

	Assigned		Actual		Emergent	
	BR	US	BR	US	BR	US
D_1	6	1	7	3	1	2
D_2	8	2	10	7	2	5
D_3	7	2	10	7	3	5
D_4	7	2	10	7	3	5
D_5	5	0	9	2	4	2
$Mean$	6.6	1.4	9.2	5.2	2.6	3.8

of the members that team members are aware of are emergent, suggesting that the ability of the SHIP's team members of staying aware of colleagues was not affected by distance.

RCSN density. Network density is defined as the proportion of ties that exist in the network out of the total possible ties. Table 6.4(a) shows the density for the communication-RCSNs. One can see that the densities are quite low, specially the Requirements negotiation networks. No network has present more than 23% out of the total possible ties (RC_{D3}), meaning that the communication that took place within each network was not higher than one-fourth of the possible communication load that the requirements-centric teams could handle. Since the emergent members are representative of the collaboration within the requirements-centric social networks (about one-third of the people involved across all networks in average), and the majority of them did not participate in the research thus they did not report who they communicated with, I thought it was important to investigate whether the network densities were affected by the presence of the emergent members. Therefore, the results of the calculation of the proportion of actual ties out of the total possible ties without considering the emergent members is presented in Table 6.4(b). The density in average doubled for each network. This indicates that team members assigned to work on the requirements communicated more in comparison to the amount of possible communication that could take place than initially suggested by the indexes in Table 6.4(a). Even though the numbers are higher, in average communication within the networks was not more than one-third of the possible communication. It is difficult to conclude whether these indexes are optimal or deficient without having any performance or quality metric to compare them against. However, since the project succeeded by deploying the requirements on time and on budget, one can conclude

that this communication was sufficient.

Table 6.4: SHIP's communication-RCSNs density

(a) With emergent members

Dep	RN	RC	CC	CA
D_1	0.08	0.22	0.17	0.21
D_2	0.08	0.20	0.16	0.15
D_3	0.09	0.23	0.18	0.16
D_4	0.08	0.20	0.16	0.16
D_5	0.07	0.20	0.15	0.15
$Mean$	0.08	0.21	0.16	0.16

(b) Without emergent members

Dep	RN	RC	CC	CA
D_1	0.07	0.50	0.40	0.36
D_2	0.09	0.31	0.31	0.26
D_3	0.10	0.39	0.39	0.32
D_4	0.10	0.36	0.37	0.32
D_5	0	0.50	0.35	0.30
$Mean$	0.07	0.41	0.36	0.31

Following the same rationale, Table 6.5(a) presents the density indexes for the awareness-RCSNs with the presence of the emergent members and Table 6.5(b) shows the densities without considering these members. One can see that in average less than 30% of the team members have knowledge of what their colleagues are doing that is related to their work when emergent people are taken into account. However, the awareness densities increase to about two-thirds of the team members when emergent people are discarded. This indicates that most of the requirements-centric team members have current awareness of their colleagues.

Table 6.5: SHIP's current awareness-RCSNs density

(a) With emergent members

Dep	Awareness
D_1	0.41
D_2	0.25
D_3	0.24
D_4	0.24
D_5	0.24
$Mean$	0.27

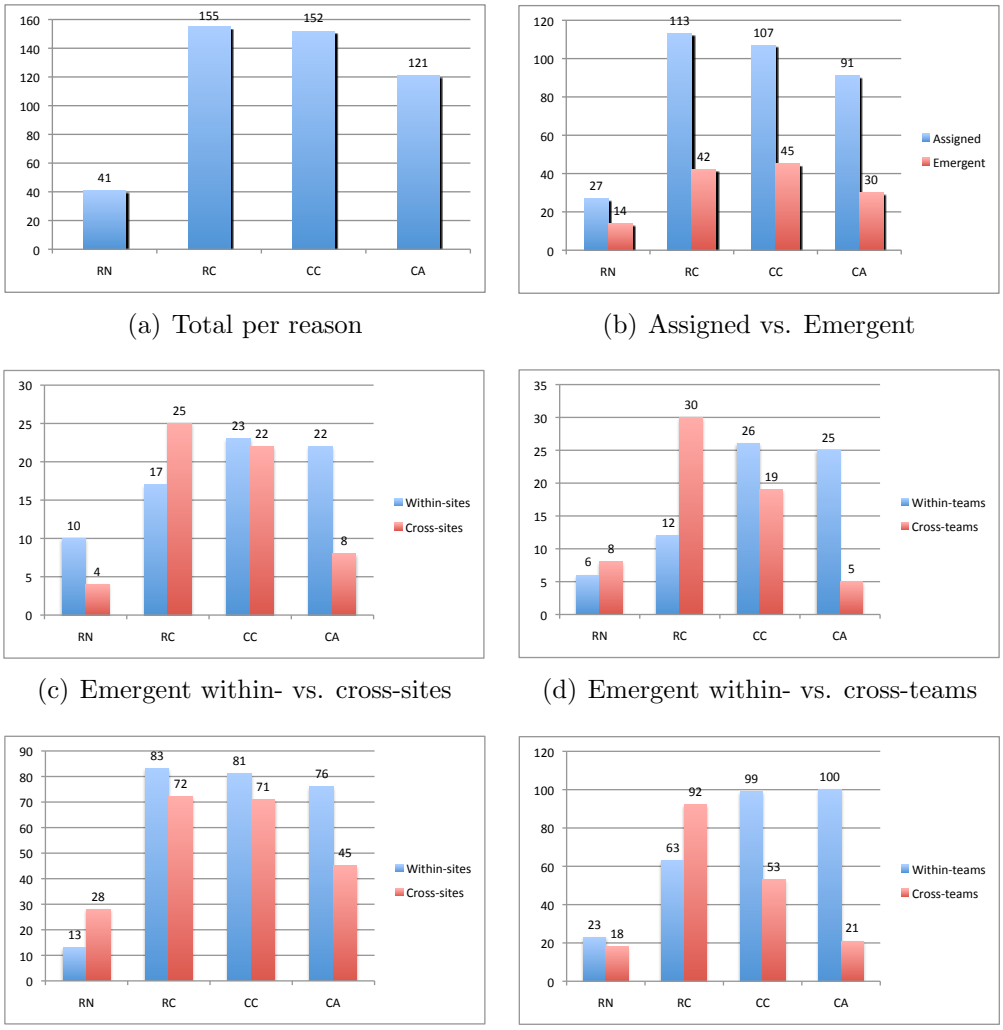
(b) Without emergent members

Dep	Awareness
D_1	0.74
D_2	0.63
D_3	0.73
D_4	0.71
D_5	0.80
$Mean$	0.72

Ties statistics based on team members attributes. To identify patterns of collaboration and information exchange within RCSNs, I provide descriptive statistics about the relationships established by the members based on their attributes. For both the communication- and the awareness-RCSNs, I counted the ties by requirement dependency set in terms of which members are involved (assigned or emergent

members), what is the location of the members (within- or cross-sites), and team membership (within- or cross-teams).

Communication-RCSNs. Figure 6.4 shows the consolidated results across the five sets of dependencies per reason of communication for simplicity. Appendix B presents the ties statistics per individual set of dependency.



(a) Total per reason

(b) Assigned vs. Emergent

(c) Emergent within- vs. cross-sites

(d) Emergent within- vs. cross-teams

(e) Within- vs. Cross-sites

(f) Within- vs. Cross-teams

Figure 6.4: SHIP's communication-RCSN across dependency sets ties statistics

In Figure 6.4(a) one can see that a total of 41 communication interactions about requirements negotiation (RN) were reported, as well as 155 interactions about requirements clarification (RC), 152 about communication of changes (CC), and 121 about coordination of activities (CA) were reported by the respondents, summing up

a total of 469 interactions. From this figure one see that requirements clarification and communication of changes are the two-most-often-reported reasons for communicating with colleagues.

Figure 6.4(b) presents the number of times per reason of communication an reported interaction took place between members who were assigned to work on the requirements (assigned) and with an emergent member (emergent). A total of 338 (out of 469, 72%) interactions were reported between assigned members and 131 (28%) with emergent members. This indicates that about one-third of the development work involves communication with people who were not assigned to work on the requirements, suggesting that project members seek a large amount of information from outsiders. From Figure 6.4(b), one see that the most-often-reported interactions between assigned members was about requirements-clarification (113/338), and the most frequent interaction with emergent members was about communication of changes (45/131), closely followed by communication about requirements clarification (42/131). It is surprising that the assigned members have communicated changes to people not assigned to work on the requirements.

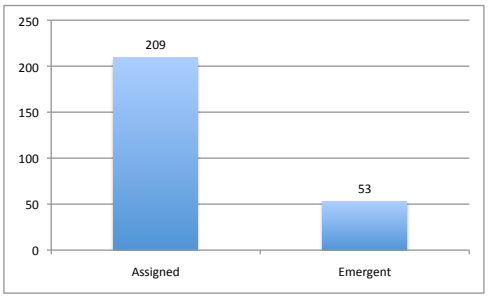
Of the 131 instances of communication with emergent members across the four reasons, in Figure 6.4(c) one can see that 54.96% were within-sites (72 instances). This indicates that local interactions have more tendency to emerge than remote interactions, suggesting that project member seeks information more readily from a local colleague. In addition, Figure 6.4(d) shows that 52.67% of the emergent communication interactions were within-teams (69 instances). Although emergent communication within-team members were slightly higher reported than communication cross-teams, this result suggests that team members are more likely to get help from people who were not assigned to work on the requirements that belong to the same team than one belongs to.

Figure 6.4(e) presents communication per reason characterized by location. Within-sites indicates communication between two members at the same location, and across-sites indicates communication between two members at different locations. A total of 253 (out of 469, 53.94%) interactions were reported between members in the same site and 216 (46.06%) between members at different locations. Although there is no significant difference between the percentages of communication by location, considering that there are less members in the US than in Brazil, the amount of cross-sites communication is rather high. This suggests that distance did not stop team members of collaborating with those in remote locations. Looking at the data in Figure

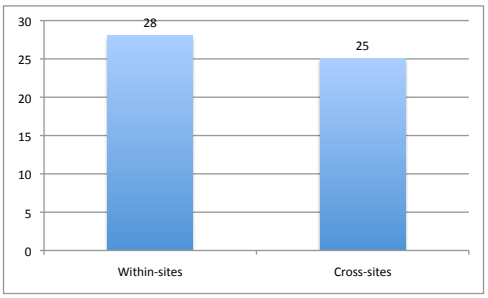
6.4(e), one see that requirements clarification (83/253) and communication of changes (81/253) are almost equally the most frequent reasons why members communicated with collocated colleagues. The same trend is observed for cross-sites communication. There were 72 (out of 216) interactions about requirements clarifications and 71 (out of 216) about communication of changes with remote colleagues. Members located in the US are either senior members with expertise on the product (development leader and a senior developer) or responsible for interacting with business partners to capture and define requirements (development leader), thus it is not surprising that members in Brazil sought their US colleagues' help to clarify requirements or were notified by the US members about changes once it is likely that changes are originated by the business partners also located in the US.

Figure 6.4(f) presents communication per reason characterized by team. Within-teams indicates communication between two members who belong to the same team (e.g., a development leader and a developer, the test leader and a tester), and across-sites indicates communication between two members who belong to two different teams (e.g., a development leader and a tester). A total of 285 (out of 469, 60.77%) interactions were reported between members belonging to the same team and 184 (46.06%) between members belonging to different teams. The high percentage of cross-teams communication suggests that the teams are well integrated and collaborate with each other to accomplish the project goals.

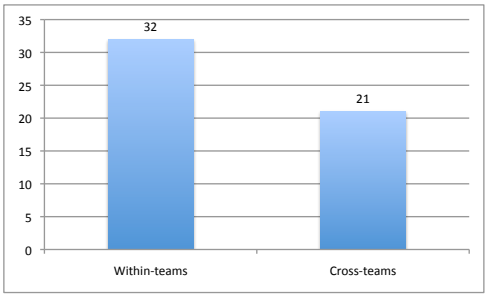
Interesting patterns are revealed when looking at the communications per reason in Figure 6.4(f). Requirements negotiation is almost equally distributes between within- (23 out of 41) and cross-teams (18 out of 41), which is comprehensive due to the nature of negotiations. It involves stakeholders who have different interests and responsibilities regarding the project, who are the business partners (represented by a business analyst), project manager, logistic manager, development leaders, and test leader. About three-fifths of the interactions about requirements clarification were cross-teams (92 out of 155, 59.35%). A more in-depth analysis showed that most of these interactions were initiated by test members and took place with colleagues from the development team. Testers were concerned in clearly comprehending the project requirements in order to ensure that they were correctly validating and verifying the product against the requirements specifications. Communication of changes (99 out of 152, 65.13%) and coordination of activities (100 out of 121, 82.64%) were predominantly discussed within teammates. This is not surprising since teammates are expected to collaborate in order to maintain synchronized information among them



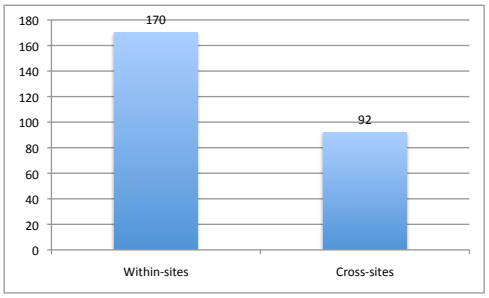
(a) Assigned vs. Emergent



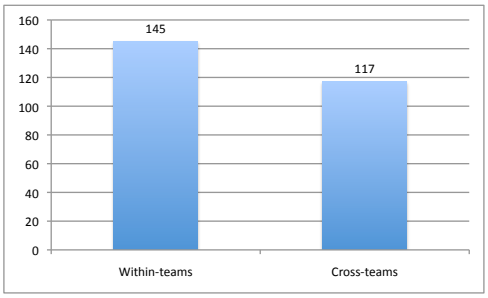
(b) Emergent within- vs. cross-sites



(c) Emergent within- vs. cross-teams



(d) Within- vs. Cross-sites



(e) Within- vs. Cross-teams

Figure 6.5: SHIP's current awareness-RCSN across dependency sets ties statistics

aiming to properly perform their work.

Current awareness-RCSNs. Figure 6.5 shows the consolidated results for the current awareness among team members across the five sets of dependencies for simplicity. Appendix B presents the ties statistics per individual set of dependency. A total of 262 instances of awareness were reported. Of these, in Figure 6.5(a) one see that 209 instances (79.77%) represent awareness between members assigned to work on the requirements (assigned) and 53 instances (20.23%) represent awareness of emergent members (emergent). This indicates that not only almost all members of the requirements-centric teams are aware of their colleagues but also that this awareness

is extended for those who emerged throughout the project life-cycle.

Of the 53 instances of reported awareness of emergent members, in Figure 6.5(b) one can see that 28 instances (52.83%) represent awareness of members working on the same site and 25 represent awareness of remote members (47.17%). This indicates that awareness of local members are more likely to emergent than of remote members. In addition, Figure 6.5(c) shows that 32 instances (60.38%) represent awareness of teammates and 21 represent awareness of members working in other teams (39.62%). This indicates that awareness of teammates is almost twice more likely to emergent than of members of other teams.

Figure 6.5(d) shows that awareness of collocated colleagues (170 out of 262, 64.89%) is about twice higher than awareness of remote members (92 out of 262, 35.11%). This reveals that team members are more knowledgeable of what people working in the same location are doing than with what remote colleagues are working on. This suggests that distance may impact the ability of team members of keeping track of remote people's work. Figure 6.5(e) also shows that team members are more aware of what their teammates are doing (145 out of 262, 55.35%) than what people in other teams are working on (117 out of 262, 44.66%). This result suggests that there is little exchange of information among team members about what colleagues outside one's team are working on.

6.2.2 (RQ3) RCSNs Structures

In order to identify what structures communication- and current awareness-RCSNs have in requirements-driven collaboration, I used the network centralization measure defined in the initial version of the proposed framework. Supplementing this single measure, I added the following measures. The *core-periphery* test, which indicates the extent to which the structure of a network consists of a core of members connected to each other, and of peripheral members who are loosely connected to the core. This test reveals to what extent requirements-driven collaboration is centered in a group of members, and who are these members. It also identifies members who are marginal in requirements-driven collaboration. The *ties reciprocity* measure which reveals the percentage of pairs of actors that are involved in a reciprocal relationship or the percentage of ties that have a reciprocated tie. Networks with high percentage of reciprocal ties are likely to be more stable and to have members mutually exchanging knowledge. The *clique* measure, which consists of a subset of at least three actors of

Table 6.6: SHIP's communication-RCSN centralization

(a) Outdegree

Dep	RN	RC	CC	CA
D_1	0.47	0.35	0.40	0.26
D_2	0.60	0.53	0.40	0.53
D_3	0.67	0.56	0.42	0.57
D_4	0.57	0.45	0.40	0.57
D_5	0.24	0.39	0.44	0.25
$Mean$	0.51	0.45	0.41	0.43

(b) Indegree

Dep	RN	RC	CC	CA
D_1	0.19	0.26	0.21	0.26
D_2	0.11	0.20	0.26	0.15
D_3	0.12	0.20	0.26	0.15
D_4	0.13	0.12	0.26	0.15
D_5	0.24	0.27	0.20	0.11
$Mean$	0.16	0.21	0.24	0.16

a network in which every possible pair of actors is directly connected by a tie and this clique is not contained in any other clique. This measure identifies which members group together to communicate about the requirements. These subgroups suggest a division within the requirements-centric teams.

RCSN centralization. A centralized network structure (index towards value 1) will have many of its ties dispersed around one or a few nodes, while a decentralized network structure (index towards value 0) is one in which there is little variation between the number of ties each node possesses. Table 6.6(a) shows the degree of centralization of communication ties sent to others, called outdegree centralization. There is no strong pattern across the reasons of communication or the sets of dependencies. Also, no network has an outdegree centralization index close to 1, except RN_{D4} index (0.67) that stands out among the others. However, one can see that 12 out of the 20 networks have index lower than 0.50, indicating that information was more equally distributed by more team members than distribution was concentrated in the hands of one or few people. This finding suggests that in SHIP the distribution of information is more decentralized than controlled by few team members.

The indegree index, presented in Table 6.6(b), refer to the degree of centralization of communication ties received from others. All networks have low indexes, varying from 0.11 to 0.27. Requirements negotiation and Coordination of activities have indexes particularly lower than the Requirements clarification and the Communication of changes reasons. These low indexes indicate that SHIP has very decentralized indegree network structures. One can then affirm that information is more equally distributed to all team members within the requirements-centric social networks.

Table 6.7(a) shows the outdegree centralization index for the awareness-RCSNS. One can see that there is a balanced pattern for how awareness of others is distributed.

Table 6.7: SHIP's current awareness-RCSN centralization

(a) Outdegree

Dep	Awareness
D_1	0.65
D_2	0.34
D_3	0.34
D_4	0.47
D_5	0.73
$Mean$	0.50

(b) Indegree

Dep	Awareness
D_1	0.28
D_2	0.27
D_3	0.21
D_4	0.21
D_5	0.18
$Mean$	0.23

$Awareness_{D1}$ and $Awareness_{D5}$ networks have higher indexes, indicating that few people are aware of many of their colleagues. $Awareness_{D2}$ and $Awareness_{D3}$ networks have low indexes, indicating that more team members are more equally aware of others. One can say that $Awareness_{D4}$ is neutral. The mean among the five networks tells one that SHIP has neither centralized or decentralized outdegree awareness-RCSNs structures.

In contrast, the indegree centralization index for the awareness-RCSNs presented in Table 6.7(b). This indicates awareness that others have of one, shows low indexes. It suggests that it is more equally distributed the awareness that others have of one. From these results one can conclude that the indegree awareness-RCSNs are decentralized, suggesting that more members of the requirements-centric teams in the SHIP project are more aware of what others are doing that is related to one's work.

Core-periphery. An additional measure to identify the network structure is the core-periphery test. This test indicates the extent to which the structure of a network consists of two classes of nodes: the core, in which nodes are connected to each other in some maximal sense, and the periphery, where nodes are more loosely connected to the core. A value close to 1 indicates a strong core-periphery structure. Table 6.8(a) shows the values for the core-periphery test for the communication-RCSNs. Three-fourths of the networks have low indexes that range from 0.13 to 0.61, exceptions are RN_{D1}, and the CC_{D1} to CC_{D4} networks. These low indexes indicate that a structure of a core and a surrounding periphery is not predominant in the SHIP communication requirements-centric social networks. This finding supports the network centralization results where it was found decentralized structures for the communication networks.

By inspecting the sociogram of the five exceptions (refer to Appendix B), which

Table 6.8: SHIP's RCSN core-periphery index

(a) Communication

Dep	RN	RC	CC	CA
D_1	0.81	0.49	0.70	0.25
D_2	0.61	0.61	0.72	0.17
D_3	0.60	0.43	0.71	0.13
D_4	0.60	0.61	0.77	0.21
D_5	0.38	0.45	0.35	0.30
$Mean$	0.60	0.52	0.65	0.21

(b) Current awareness

Dep	Awareness
D_1	0.51
D_2	0.72
D_3	0.74
D_4	0.72
D_5	0.95
$Mean$	0.73

have indexes higher than 0.71, one can identify that a set of members form a core structure. The core-periphery test lists which members are part of the core and the periphery classes, respectively. The members in the core for each network are as follows.

- RN_{D1}: AC (Dev Lead), and EZ (Dev Lead)
- CC_{D1}: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), DS (Developer), and AM (Tester)
- CC_{D2}: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), LL (Developer), PR (Developer), and AM (Tester)
- CC_{D3}: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), LL (Developer), PR (Developer), and AM (Tester)
- CC_{D4}: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), LL (Developer), PR (Developer), and AM (Tester)

It is interesting to note that the three leaders in the project are present in the core of the five networks, except the test leader (named LS) in the RN_{D1}. Also, two of the most senior developers participate in the core of three Coordination of activities networks, namely LL (Developer) and PR (Developer). LL is a developer located in Brazil and he is the most experienced Brazilian team member after the development leader. He travelled to work collocated with the US team twice in the three months period I spent on site. The development leader wanted him working closely to the ORG shipping partner to facilitate the understanding of the technical challenges the team have to go through to integrate ORG and the partner's systems.

PR is a developer located in the US and she is the second oldest current member working in the SHIP project, following the US development leader. PR is a very active developer who was working closely to LL in integrating ORG's system with the shipping carrier' system. She was aware of some business agreements between ORG and the carrier system. This helped the team to speed up the development of the functional specifications for the requirements that were related to the carrier' system. PR also works close to the US development leader, *AC*, and in past projects she acted as an interim development leader while AC was busy negotiating requirements with business partners.

It is surprising that the member who last joined the team, AM (tester), is present in the core of three of the five networks. He started working in the SHIP project about six weeks before I started my field visit, and he had just completed his company on boarding training and was still waiting for the test leader to tell him he was prepared to work without a mentor supervising him. Because he was attending all the meetings and interacting with as most colleagues as he could to learn about his work, he may have got involved more actively with communicating about the project than expected.

In contrast to the communication-RCSNs, the awareness-RCSNs have high indexes suggesting that the networks have a more-like core-periphery structure. $Awareness_{D1}$ is an exception. Surprisingly, this results somehow do not support the network centralization results for the awareness networks, specially the indegree centralization network indexes that suggest a decentralized structure. A look at the list of members that compose the core of the awareness-RCSNs explains the discrepancy between the network centralization and the core-perihery test results (refer to Appendix B for the complete list). All members identified by the test as peripheral members, with exception of two (out of 31), are emergent members. Members of the requirements-centric teams indicated they were aware of them, but in fact the emergent members were not assigned to work on the requirements, thus they had no obligation to be aware of the colleagues. This finer-grain examination of these results reveal that the assigned members of each awareness-RCSNs belong to the network core. It is possible that the core-periphery test would yield a result supporting the network centralization finding if the emergent members were removed from the analysis. The sociograms of the awareness networks suggest that it is likely this supposition is confirmed. When visually examining the networks, the perception is that the ties are equally distributed among the members, potentially confirming the decentralized structures pointed out by the network centralization measure.

Ties reciprocity. Another additional measure to the initial set of measures proposed in the framework to examine the network structures is the ties reciprocity measure. In a directed network, one can examine the percentage of pairs of actors that are involved in a reciprocal relationship (dyad-based method) or the percentage of ties that have a reciprocated tie (arc-based method). Table 6.9(a) and Table 6.9(b) present the reciprocity percentages for each method, respectively. For both methods the percentages are low. I visually inspected the networks aiming to grasp whether reciprocity have been affected by the presence of the emergent members, and I noticed that the percentages would still be low if emergent members had been removed. Thus, overall, results suggest that communication is not mutually exchanged in the requirements-centric social networks. A more fine-grained analysis of which pair of members are involved in reciprocal communication would reveal whether communication is one-way because of the role that the members play in the project (e.g., a development leader may be expected to communicate changes to developers but not necessarily developers have to return the notification).

Despite the low indexes, when comparing one method to another, one can see that the percentages in the arc method are higher than the percentages calculated by the dyad method. This indicates that there are more pairs of reciprocal communication interactions across the networks than pairs of team members who got involved in reciprocal relationships. More specifically, one can observe that the Requirements negotiation reason has the lowest levels of reciprocities between pairs of team members, meaning that negotiation was not a two-ways process between members in this project. Communication of changes is the reason that most have reciprocal ties, suggesting that when a team member notified a colleague of a change the colleague communicated back. It is likely this reciprocal communication intended to confirm awareness of the change, or initiated a communication thread about the change itself.

Table 6.9: SHIP's communication-RCSN ties reciprocity

(a) Dyad-based method

Dep	RN	RC	CC	CA
D_1	0.20	0.26	0.44	0.36
D_2	0.22	0.20	0.31	0.19
D_3	0.25	0.24	0.31	0.25
D_4	0	0.24	0.36	0.25
D_5	0	0.20	0.17	0.10
$Mean$	0.13	0.23	0.32	0.23

(b) Arc-based method

Dep	RN	RC	CC	CA
D_1	0.33	0.41	0.61	0.53
D_2	0.36	0.33	0.47	0.32
D_3	0.40	0.39	0.47	0.40
D_4	0	0.39	0.53	0.40
D_5	0	0.33	0.28	0.18
$Mean$	0.22	0.37	0.47	0.36

Table 6.10: SHIP's awareness-RCSN ties reciprocity

(a) Dyad-based method

Dep	Awareness
D_1	0.61
D_2	0.46
D_3	0.44
D_4	0.43
D_5	0.30
$Mean$	0.45

(b) Arc-based method

Dep	Awareness
D_1	0.76
D_2	0.63
D_3	0.61
D_4	0.60
D_5	0.46
$Mean$	0.61

The pattern is repeated for the awareness-RCSNs. The arc-base method percentages are higher than the dyad-based method, indicating that there are more awareness reciprocity between members than pairs of team members that are aware of each other. A graphical inspection of the awareness networks reveals that if the emergent members were to be removed, most of the relationships would be reciprocal. As a consequence, higher arc-based reciprocity indexes would be presented. This means that in fact the assigned members are aware of each other as expected in a group of people working together aiming to accomplish a task. It is challenging to define whether the dyad-base method would yield higher indexes by visually inspecting the networks, thus I prefer to not suggest any change in behavior for this method. By contrasting Table 6.9 with Table 6.10 one can see that reciprocity is higher in the awareness networks than in the communication ones. This suggests that requirements-centric team members are more mutually aware of each other than they mutually communicate with each other.

Clique. Not initially included in the proposed framework, a clique consists of a subset of at least three actors of a network in which every possible pair of actors is directly connected by a tie and this clique is not contained in any other clique. In directed networks there are two types of cliques: the strong, where only reciprocal ties between pairs of actors are considered, and the weak, when the direction of the tie is disregarded and simply the presence or absence of a relation is considered. Since no strong clique could be computed for most of the networks Table 6.11 presents the number of weak cliques found for the communication-RCSNs. No pattern across the reasons of communication or the set of dependencies stands out. However, one can see that Requirements negotiation is the reason of communication with the small-

Table 6.11: SHIP's communication-RCSN weak cliques

Dep	RN	RC	CC	CA
D_1	1	5	3	5
D_2	2	9	6	10
D_3	1	8	6	9
D_4	1	7	5	6
D_5	0	3	1	2
Sum	5	32	21	32

est number of cliques (total of 5 across the networks). Requirements clarifications and Coordination of activities have the highest numbers of cliques (32 in total each reason). This high number of cliques within these two reasons of communication suggests that there are well-connected subgroups of team members, potentially forming informal structures within the networks. A more detailed investigation of whose members consist these cliques was then conducted based on the list of members that compose each clique provided by the measure calculation aiming to identify details about these structures. The distribution of cliques for each reason of communication by roles involved in a single clique is as follows. Appendix B lists the team members' names by clique.

- RN: leaders and manager (2); leaders and senior devs (3)
- RC: leaders and testers (11); leaders and senior devs (4); leaders and senior dev and tester (3); test team and business analyst (4); test team and logistic manager (3); test team and senior dev (4); dev team (3)
- CC: leaders and devs (9); leaders and tester (1); dev team (8); test team (3)
- CA: dev team (16); test team (5); leaders and senior dev (6); test team and senior dev (3); test leader and senior dev (2)

This in-depth analysis of which members are involved in the identified cliques reveals interesting grouping behavior. In the Requirements negotiation networks one can see that cliques were formed by the project leaders (development leaders and the test leader) and the project manager, and by the leaders and senior developers. These members are the ones responsible for negotiating the requirements with business partners, except by the senior developers who for having knowledge of the business domain from past project releases ended up supporting the development leaders during the

negotiations. This results only reinforces that the team actual collaboration behavior followed the team processes and respected the roles' responsibilities. The majority of the cliques formed in the Requirements clarification reason involved members of the test team. During the observation sessions on site I noticed that the test team had a special concern in ensuring the understanding of the project requirements. The creation of testing requirements, which represent a finer-grain level specification of the project requirements, is a demonstration of the effort put by the test team in the comprehension of the requirements. It is interesting to note that test team members were well-connected not only to leaders and senior developers but also to the business analysis and the logistic manager, who were emergent members within these requirements-centric social networks. These results suggest that the involvement of people with business expertise in the project is important to ensure that the team has a clear and common understanding about the requirements.

Although most of the cliques within the Communication of changes RCSNs involve members of the development team, it is interesting to note that cliques were formed among members in the same team (development and test teams) and among leaders and developers, and leaders and testers. These results indicate that notifications of changes have been closely propagated and discussed among those who negotiate requirements (leaders) and those who implement them (developers and testers). Another interesting finding is that team members grouped together to coordinate activities (CA reason) with members of their own teams or with senior developers. Coordinating activities with teammates is an expected collaboration pattern confirmed by this in-depth analysis but the formation of well-connected groups involving senior developers in the coordination of activities is somehow surprising. One would expect that leaders, or at least the development leaders, would be the members involved given their broad view of the project activities and requirements dependencies.

From this detailed analysis of the cliques compositions, it is possible to conclude that the informal structures formed through the cliques are organized around organizational teams (development and test) or are motivated by the need of obtaining information from experts about the project and the product that is maintained (leaders or senior developers).

6.2.3 (RQ4) RCSNs Information Flow

To identify specific patterns of information exchange and knowledge sharing, I applied the component, degree centrality, and brokerage measures defined in the proposed framework. I added to other measures to this initial set, which are reachability and cutpoint. The *reachability* measure defines that an actor is reachable by another actor if exists any set of ties that connects both actors, regardless of how many others fall between them. This measures indicates the ability that each requirements-centric team member has to reach other members, and of being reached by colleagues of the requirements-centric team. The *cutpoint* measure identifies an actor (or set of actors) that if removed along with his ties the network would become divided into unconnected parts. This measure reveals critical members for the continuance of information exchange and knowledge sharing. Flow would be disrupted if these members were removed from the RCSNs.

Component. The component test indicates whether a social network is connected. The network is connected if there is a path between every pair of nodes, otherwise it is disconnected. The connected subsets in a social network are called components. For directed networks, two kinds of components can be defined: a weak component, where a set of actors are connected regardless of the direction of ties, and a strong component, where a directed path from actor A to actor B is expected for the two actors be in the same component. Table 6.12(a) shows the amount of components found when the direction of the ties are ignored. One can see that each network has a single component, meaning that there are no isolated members, except by the Requirements negotiation networks. The multiple components within each Requirements negotiation RCSNs indicate that members assigned to work in these networks did not get participate in the negotiation of the project requirements. By visually inspecting the sociograms (refer to Appendix B) one can observe that isolated members in these networks are either developers or testers, roles that are not responsible for negotiating requirements.

On the other hand, when one considers the direction of the ties the networks are divided in multiple components as showed in Table 6.12(b). This division is characterized by several components of one member only, i.e. isolated members, and a larger component with the remaining members. In practical terms, since the networks report communication that did take place in a real situation, one can conclude that

Table 6.12: SHIP's communication-RCSN component

(a) Weak component

Dep	RN	RC	CC	CA
D_1	5	1	1	1
D_2	5	1	1	1
D_3	4	1	1	1
D_4	4	1	1	1
D_5	4	1	1	1

(b) Strong component

Dep	RN	RC	CC	CA
D_1	8	5	6	4
D_2	9	8	9	8
D_3	8	7	8	6
D_4	11	8	10	7
D_5	8	6	7	7

the components revealed by the strong component measure indicates that although some members have communicated with colleagues (because the weak component results did not indicate isolates for the majority of the networks), there are members in each network that did not try to reach their requirements-centric team colleagues. In fact, a closer look in the members that compose each strong component (refer to Appendix B) reveals that for the majority of the networks the members that did not try to contact others (there is no path between them and others) are the emergent members. Like in the core-periphery test where most of the members in the periphery of the awareness networks are emergent members, here the emergent members are excused for not trying to reach their colleagues since they were not assigned to work in the requirements that the requirements-centric social networks are about.

Reachability. An additional measure to identify patterns of information exchange and knowledge sharing is the reachability measure. This measure defines that an actor is reachable by another actor if exists any set of ties that connects both actors, regardless of how many others fall between them. In a directed network directed paths between actors are considered. I have used this measure as a resource to compliment the results of the component measure. Due to the large number of communication networks investigated in this research, the resulting reachability matrix for each network are presented in the Appendix 3. Here I highlight the main patterns found. Figure 6.6 shows the reachability matrix for the Requirements negotiation-RCSN for the dependency set D_1 as an example. For instance, in the Requirements negotiation networks testers cannot reach any of their requirements-centric team colleagues or can be reached by any of them. This confirms what can be identified by the visual inspection of these networks: that testers are isolated in these networks. The fact that testers are not able to reach or be reached is probably not problematic since

	J	A	A	E	D	L	S	B	M
JS (Tester)	0	0	0	0	0	0	0	0	0
AM (Tester)	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	0	0	0	1	0	1	0	1	1
EZ (Dev Lead)	0	0	1	0	0	1	0	1	1
DS (Developer)	0	0	0	0	0	0	0	0	0
LS (Test Lead)	0	0	0	0	0	0	0	1	0
SB (Tester)	0	0	0	0	0	0	0	0	0
BF (Project Manager)	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0

Figure 6.6: SHIP's requirements negotiation-RCSN reachability matrix for dependency set D_1

testers are not responsible for negotiating requirements. The pattern that stands out within the Requirements clarification networks is that emergent members cannot reach any of their network colleagues. This result corroborates with the peripheral position found by the core-periphery test for and the isolation in the strong component test of the emergent members. Once again this result is not surprising since emergent members were not assigned to work on the requirements. They were invited to clarify requirements by assigned members, and thus are not expected to be able to communicate with any other requirements-centric team member.

In all Communication of changes and Coordination of activities networks emergent members cannot reach any of their colleagues. In addition the dependency set D_5 for each of these reasons presents an interesting case. The US development leader, named *AC*, is an emergent member who brings up with her two emergent members who are: *ML*, a former US development leader who still informally supports the team when necessary, and *BF* who is the actual project manager. Interestingly, she cannot reach any other member within these two networks but these two emergent colleagues, who cannot also reach anyone else. This indicates that she somehow became aware of the changes that were taking place on the D_1 set and coordinated change-related activities with both colleagues.

Cutpoint (or cutsets). Another additional measure to the proposed framework is the cutpoint measure. A cutpoint is an actor identified as a weak point in the network because if the actor is removed along with his ties, the network would become divided into unconnected parts. If there is a set of actors that fit this criteria, these actors are called a cutset. Table 6.13 presents the list of cutpoints (or cutsets) for

Table 6.13: SHIP's communication-RCSN cutpoints

Dep	RN	RC	CC	CA
D_1	AC (Dev Lead)	AC (Dev Lead) SB (Tester)	AC (Dev Lead) SB (Tester) LS (Test Lead)	AC (Dev Lead) LS (Test Lead)
D_2	AC (Dev Lead)	AC (Dev Lead)	AC (Dev Lead) SB (Tester) LS (Test Lead) PR (Developer)	AC (Dev Lead) PR (Developer)
D_3	AC (Dev Lead)	AC (Dev Lead)	AC (Dev Lead) SB (Tester) LS (Test Lead) PR (Developer)	AC (Dev Lead) PR (Developer)
D_4	AC (Dev Lead)	AC (Dev Lead)	AC (Dev Lead) SB (Tester) LS (Test Lead) PR (Developer) EZ (Dev Lead)	AC (Dev Lead) PR (Developer)
D_5	AC (Dev Lead)	AC (Dev Lead) SB (Tester)	AC (Dev Lead) SB (Tester) LS (Test Lead)	AC (Dev Lead) EZ (Dev Lead)

each communication-RCSN. Note that there is at least one cutpoint present in each network, and that the US development leader named AC is a critical member for the continuance of exchange of information in each and every of the networks. This finding is surprising despite the fact that during my on site visit I have noticed the important role that she played in the project along with the Brazilian development leader, the test leader, and two senior developers (one located in the US and one in Brazil). She was not only the most senior member of the team, with a deep knowledge of the product and the team processes, but also the focal point for interactions with the business partners, and the shipping IT teams at the ORG worldwide manufacturing plants. This result suggests that the project depends on AC and that it would collapse without the her active participation in the project. The dependency the project has on AC collaboration and development of activities dangerous for a project so critical for ORG like SHIP.

Interesting patterns are found when one looks at the cutpoints (or cutsets) across each reason of communication. Requirements negotiations and requirements clarifications networks are mainly dependent on AC only as discussed. In other words,

some team members would not be able to exchange information about requirements negotiations and also would not have access to an expert to clarify requirements if the US development leader AC suddenly became unavailable or, worse, left the project.

Communication of changes would be delayed or not propagated at all if the test leader named *LS* and the US senior developer named *PR* were excluded from the networks. The fact that the communication of changes networks depend on cutsets with several members (average of 4 members) indicates that these networks are less vulnerable of suffering disruption in the exchange of information about changes. One would have to eliminate a considerable number of members from each network to cause the flow of information to break. In practice, this result suggests that somehow team members would find ways to propagate notifications of changes to others. The Coordination of activities networks, though, have similar pattern than the two first reasons of communication. They depend on AC and either on another project leader (from the development or the test team) or on the senior developer PR.

Degree centrality. The degree centrality indicates the number of ties of an actor and is indicative of activity. In a directed network, the count is made for out-ties (outdegree), which are ties from a certain actor to others, and for in-ties (indegree), which are ties from others to a certain actor. A complete list of the degree centrality for each set of requirements dependency is presented in the Appendix 3. Table 6.14 highlights the team member (or set of members in case that the same degree was found for more than one person within a RCSN, indicated with a *) who have the highest communication degree centrality by dependency set.

Table 6.14(a) presents who most communicated with others. For the requirements negotiation reason, one see that the US development leader, named *AC* is the most active person. She was in fact the main point of contact with business partners, and she was closely supporting the project manager in making decisions about what changes request to accept and which ones were more suitable to be put in a back log for future releases. The project manager named *BF* was new to the project and was still learning his way around the team and the work to be done. The test team, represented by the test leader named *LS* and a senior tester named *SB*, was who exclusively most requested clarifications about requirements. In contrast, the development team, here represented by the US development leader named *AC* and the US senior developer named *PR*, was who predominantly most propagated notification of changes. This is somehow not surprising since these both members were in contact with the business

Table 6.14: SHIP's communication-RCSN highest degree centrality member

(a) Outdegree

Dep	RN	RC	CC	CA
D_1	AC (Dev L.)	test team*	AC (Dev L.)	AC (Dev L.)
D_2	AC (Dev L.)	LS (Test L.)	dev team*	PR (Dev.)
D_3	AC (Dev L.)	LS (Test L.)	dev team*	PR (Dev.)
D_4	AC (Dev L.)	test team*	dev team*	PR (Dev.)
D_5	AC (Dev L.)	test team*	LS (Test L.)	LS (Test L.)

(b) Indegree

Dep	RN	RC	CC	CA
D_1	BF (P. Man.)	EZ (Dev L.)	LS (Test L.)	LS (Test L.)
D_2	leaders	dev leaders*	LS (Test L.)	leaders*
D_3	leaders*	EZ (Dev L.)	LS (Test L.)	AC (Dev L.)
D_4	leaders*	dev leaders*	LS (Test L.)	LL (Dev.)
D_5	AC (Dev L.)	dev leaders*	AC (Dev L.)	leaders*

Table 6.15: SHIP's awareness-RCSN highest degree centrality member

(a) Outdegree

Dep	Awareness
D_1	EZ (Dev Lead)
D_2	test team*
D_3	test team*
D_4	EZ (Dev Lead)
D_5	EZ (Dev Lead)

(b) Indegree

Dep	Awareness
D_1	test team*
D_2	dev leaders*
D_3	dev team*
D_4	dev team*
D_5	EZ (Dev Lead)

partners, where more likely the changes were originated. A similar trend is found for coordination of activities. AC and PR were the two members who most coordinated activities with others.

The indegree results for the communication-RCSNs showed in Table 6.14(b) reveals that most of members who most received information from colleagues are leaders, who are in charge of managing and coordinating their teams work.

Table 6.15(a) shows the member (or set of members, indicated with a *) who is more aware of others for each dependency set. Note that no pattern stands out across all sets of dependencies. However, the Brazilian development leader named *EZ* is the person most aware of others for 3 out of the 5 dependencies. During my visit on site I noticed that EZ had a daily routine. After lunch he used to get a cup of coffee in the building cafeteria and walk around his development colleagues' cubicles

while drinking his coffee asking if they were on time with their tasks and if he could be of any help. This five to ten minutes he spent daily was not only his strategy to find out whether any delay or problem was about to be experienced but it also promoted a relaxing atmosphere in the work environment. He used to bring candies with him and offer to his colleagues. This cordiality and establishment of a convivial work environment is a very typical social behavior for Brazilians. But EZ did not limit himself to learn what his teammates were doing. He frequently, very often more than once a day, got engaged in short phone calls (2-3 minutes) or instant message conversations with the US development leader to talk about what was going on from the business and management perspectives, and what he could do to keep the flow of work on track.

There is also no pattern standing out in Table 6.15(b), where awareness that others have of one is presented. However, one can see that the development team represented by the development leaders and the US senior developer named *PR* is predominant as the group of people that project members most are aware of (refer to Appendix 3 for the complete list of indegree awareness per dependency set]). It is likely that this behavior is justified by the fact that the test team, mainly the test leader and a most senior tester, was constantly checking on whether the development team was time-wise on track with its activities or whether requirements have changed aiming to analyze impact of delays or changes to the test schedule and work strategy.

Brokerage. Brokerage indicates when an actor, named *broker*, connects two otherwise unconnected actors or subgroups. To calculate brokerage the actors are divided into mutually exclusive groups. Since I was interested in examining flow of communication across dependent requirements I used the assignment to a certain requirement as the attribute to divide the members into groups. Then, ten possible configurations of brokers were defined to address the following three information flows: outgoing flow, incoming flow, and consulting flow. Details about the broker configurations and the information flows has been presented in Chapter 4.

Table 6.16 shows all 44 brokers identified. Each communication-RCSN, represented by a cell in Table 6.16, can have up to ten broker types and each member can take the role of each broker. Observe that the brokers concentrate in the Requirements Clarifications, Coordination of Activities, and Communication of Changes reasons of communication. Each broker in each network is indicated by the configuration and member fictitious name (e.g., b_8:LS). Some members are associated with multiple

Table 6.16: SHIP's distribution of brokers cases per configuration

Dep	*RN*	*RC*	*CC*	*CA*
D_1	-	b_2:AC (DevL) b_5:AC (DevL) b_8:AC (DevL) b_8:LS (TestL)	b_5:EZ (DevL) b_8:LS (TestL)	b_5:EZ (DevL) b_8:LS (TestL)
D_2	b_5:AC (DevL)	b_2:AC (DevL) b_5:AC (DevL) b_6:PR (Dev) b_8:AC (DevL)	b_2:AC (DevL) b_5:AC (DevL) b_6:PR (Dev) b_6:LL (Dev) b_8:AC (DevL) b_8:EZ (DevL)	b_2:AC (DevL) b_5:LS (TestL) b_5:AC (DevL) b_6:LL (Dev) b_6:PR (Dev) b_8:EZ (DevL)
D_3	b_4:AC (DevL)	b_3:PR (Dev) b_4:AC (DevL) b_5:SB (Tester) b_8:LS (TestL) b_8:EZ (DevL)	b_3:PR (Dev) b_4:AC (DevL) b_8:LS (TestL)	b_3:PR (Dev) b_4:AC (DevL) b_5:LS (TestL) b_8:EZ (DevL) b_9:PR
D_4	-	-	-	-
D_5	b_5:AC (DevL)	b_5:AC (DevL)	b_5:LS (TestL) b_5:AC (DevL)	b_5:AC (DevL)

configurations.

Outgoing flow. This flow investigates the presence of brokers who mediate information from the dependee to the dependent requirement. Outgoing flow brokers are those who fall in one of the following configuration cases: b_1, b_2, b_4, b_5, b_7, and b_8. Out of the total of 44 instances of brokerage, 35 indicate outgoing flow in Table 6.16. They are distributed over 15 of the 20 RCSNs as follows: $(b_1) = 0$; $(b_2) = 4$; $(b_4) = 4$; $(b_5) = 15$; $(b_7) = 0$; and $(b_8) = 12$. These results indicate that most of the outgoing brokers are members that were assigned to work on both dependent requirements.

The brokers for outgoing flow are enumerated below, together with their role in the project, geographical location and number of occurrences over the total number of outgoing brokers. One particular person (AC) emerges most frequently across all networks. This person is the US-based development leader. The distribution of all brokers is as follows: LS (Test Lead, BR): 8/35; EZ (Dev Lead, BR): 6/35; AC (Dev Lead, US): 22/35; SB (Tester, BR): 1/35; and PR (Senior Dev, US): 0/35.

Incoming flow. This flow investigates the presence of brokers who mediate information from the dependent to the dependee requirement. Incoming flow brokers are those

who fall in one of the following configuration cases: b_2, b_3, b_5, b_6, b_8, and b_9. Out of the 44 cases, 40 indicate incoming flow in Table 6.16. These brokers were identified in 14 out of the 20 RCSNs, and are distributed as follows: $(b_2) = 4$; $(b_3) = 3$; $(b_5) = 15$; $(b_6) = 5$; $(b_8) = 12$; $(b_9) = 1$. Similar to the outgoing flow, these results indicate that most of the incoming brokers are project members that work on both dependent requirements.

A similar trend can be observed by examining the distribution of members in these configurations. Same US-based development leader (AC) is the most frequent broker. The distribution of all brokers is: LS (Test Leader, BR): 8/40; EZ (Dev Leader, BR): 6/40; AC (Dev Leader, US): 16/40; SB (Tester, BR): 1/40; PR (Senior Dev, US): 7/40; and LL (Senior Dev, BR): 2/40.

Consulting flow. This flow investigates the presence of brokers who mediate information either from the dependent or from the dependee requirement. Thus, consulting brokers are those who fall in one of the following configuration cases: b_4, b_9, and b_10. Out of the 44 cases, only 5 indicate consultant flow. They took place in 4 out of the 20 networks, and are distributed as follows: $(b_4) = 4$; $(b_9) = 1$; $(b_10) = 0$.

The distribution of brokers for the consulting flow is as follows: AC (Dev Leader, US): 4/5 and PR (Senior Dev, US): 1/5. Again, AC is the dominant person. AC and PR are in US and are very active with Brazil. AC mediated twice between Brazil members and twice between Brazil and US; PR mediated between members located in Brazil.

The presence of brokers in any of these networks indicates the existence of pairs across two dependent requirement networks that do not communicate directly and for which the information flow was mediated. In the context of requirements engineering, this indicates that those members whose communication was brokered did not use and possible had no direct way of communicating requirements information. Misunderstandings or information loss could occur in these mediated interactions. Note that information on the flow of a unit of information from sender to broker and then from broker to receiver were not collected. Instead, I analyzed the flow of information in general between members, without knowing if the same specific unit of information was transferred. This limitation is minimized by the fact that respondents reported which topic they were discussing with colleagues, thus information brokered by the identified brokers is likely to be communicating about the same subject.

6.2.4 (RQ5) RCSNs Alignment

To examined the alignment between RCSNs I used the current socio-technical congruence measure and the Role-based measure as presented in the initial version of the framework. No additional measures were defined.

Alignment of networks. In order to examine to what extent requirements-driven collaboration patterns are aligned with coordination needs established by requirements dependencies, I used the socio-technical congruence measure proposed by Cataldo et al. [24]. This measure calculates the ratio of actual social interactions over the expected coordination needs defined based on the technical dependencies in the project (refer to Chapter 2 for details).

Requirements-driven collaboration patterns may be affected by imposed organizational structures. Then, in addition, I extended the current socio-technical congruence measure to investigate to what extent requirements-driven collaboration patterns are aligned with the organizational structure defined by the communication channels established between pairs of roles. The extended measure has been named *Role-based socio-technical congruence* measure, and it was presented in Chapter 4. The application of this extended measure allowed the identification of false gaps and backchannel communication. Next I present the results found by both measures, constantly contrasting one against another.

Calculating Cataldo's congruence. For each measure, I calculated coordination needs and actual coordination according to the definitions presented in Chapter 2 and Chapter 4. I will refer here to each definition and equation presented in these chapters for simplicity. To calculate Cataldo congruence index (Equation 2.2), I computed the coordination needs matrix followed by the actual coordination matrix. To compute the *coordination needs matrix* (Equation 2.1), I constructed the *tasks assignment* and *tasks dependency matrices* from the requirements dependencies and team members data gathered. Since my unit of analysis is a requirement, I will rename the "tasks assignment" matrix and "tasks dependency" matrix to *requirements assignment* matrix and *requirements dependency* matrix, respectively. I computed 20 *actual communication matrices* for SHIP from the reported questionnaire communication interactions data. Each matrix represent coordination activities that took place for a certain set of requirements dependency and were about a certain reason. Remember that there are 5 sets of dependencies for SHIP, and that I asked team members

to report on 4 reasons of communication. For example, the SHIP D_1com_{RN} actual coordination network captures requirements negotiation communication among team members for the set of requirements that belong to the dependency D_1. I calculated a *lack-of-coordination matrix* (Equation 2.3) to identify the congruence gaps by each combination of each set of dependencies and each reason of communication. For each gap instance, I gathered demographic information on the pair of members involved on the gap. In addition to the team member role, demographic information include the team member office location, time working in the project, time working in the company, and the work relationship (employee or contractor).

Calculating the Role-based congruence. As presented in Chapter 4, to calculate the Role-based congruence index (Equation 4.4), I computed the prime coordination needs matrix followed by the alignment coordination matrix. To compute the *prime coordination needs matrix* (Equation 4.2), I used the *requirements assignment matrix* and *requirements dependency matrix* previously constructed and constructed two other matrices. I constructed the *role-role matrix* from the organization' structure data and the *role-assignment matrix* as described in Chapter 4. Then, I calculated the *role-coordination* matrix (Equation 7.2.4). To compute the *alignment coordination matrix* (Equation 4.3), I used the *actual communication matrices* previously constructed and the prime coordination needs matrix. I then removed all cases where a role is not allowed to communicate with another role as defined in Equation 4.3. I calculated 20 alignment coordination matrices for SHIP, each representing coordination activities that took place for a certain set of requirements dependency and due to a certain reason like explained for the actual coordination matrices in Cataldo congruence.

I calculated a *prime lack-of-coordination matrix* (Equation 4.5) to identify the real gaps by each combination of each set of dependencies and each reason of communication. I also identified false gaps (Equation 4.6) by subtracting each cell in the prime lack-of-coordination matrix from the role-coordination matrix (Equation). I finally calculated the backchannel communication (Equation 4.7). For each real gap, false gap, and backchannel communication instance, I gathered the same set of demographic information on the pair of members as for the gaps identified in Cataldo congruence.

Below I present the results of the application of the Cataldo and of the Role-based socio-technical congruence measures for the SHIP project. The results are, alter-

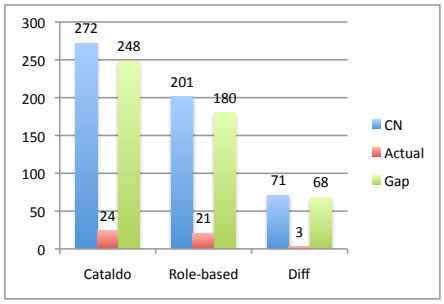
nately, grouped by reason of communication, sets of dependencies, or roles performed by the team members. Differences between results obtained by applying Cataldo's and the Role-based measure are highlighted throughout the description of the results.

Coordination needs, actual communication, and gap numbers. I calculated the coordination needs, identified the actual (and aligned) communication, and computed the amount of congruence gaps for each set of dependent requirements network by reason of communication as previously explained. In Figure 6.7 I present consolidated results across the 5 sets of dependent requirements by reason of communication. For instance, there are 272 coordination needs in this project as shown in Figure 6.7. The first cluster of columns from the left to the right side indicates the coordination needs (CN), actual communication (Actual), and gaps (Gap) calculated by Cataldo measure. The second cluster of columns indicates the coordination needs, aligned communication, and real gaps calculated by the Role-based measure. The third cluster indicates the difference between Cataldo's and the Role-based's coordination needs, the amount of backchannel communication, and the amount of false gaps across the 5 sets of dependencies.

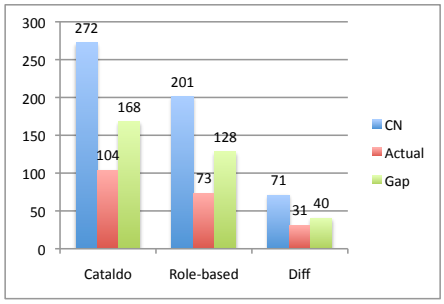
Overall, the low amount of backchannel communication showed in Figure 6.7 indicates that the requirements dependencies created certain coordination needs that were mostly attended in alignment with the imposed organization structure. The communication channels made available by the organization were almost sufficient for the team to coordinate work necessary to accomplish the project goals. On the other hand, the low amount of false gaps showed in Figure 6.7 indicates that the few of the lack of actual communication for certain coordination needs were between pairs of roles who were not expected to directly communicate with each other. Mostly the gaps are between pairs that could have communicated with each other (real gaps). This finding suggests the imposed organization structure is not the reason why people did not coordinate work.

Socio-technical congruence index. The level of alignment between the coordination needs and actual coordination was calculated for each set of dependent requirements and reason of communication network as presented above. Table 6.17(a) and Table 6.17(b) display, respectively, the social-technical congruence indexes obtained by applying Cataldo and the Role-based measure. Although the range of variation of the indexes is similar for both measures, they vary from 0 to 53% by Cataldo

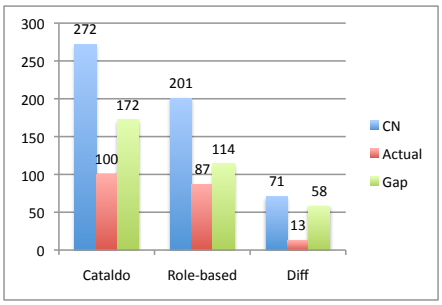
Figure 6.7: SHIP's coordination needs, actual coordination, and gap details



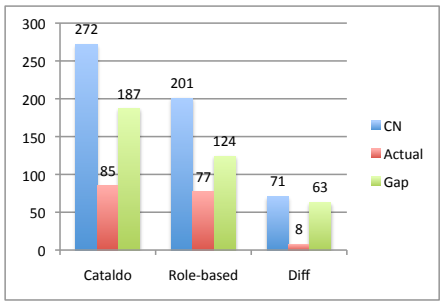
(a) Requirements Negotiation



(b) Requirements Clarification



(c) Communication of Changes



(d) Coordination of Activities

and from 0 to 52% by the Role-based measure, individually most of the indexes are slightly higher in the Role-base measure calculation. The Requirements clarification indexes for the dependency sets D_1 and D_3-D_5 are an exception. The Requirements negotiation-D_5 index has not changed. These indexes indicate that, on average, no more than about half of the coordination needs were satisfied per network.

Congruence Gaps by Roles. I further the investigation of requirements-driven coordination behavior by exploring demographic data about the identified gaps. I break down the gap information presented in Figure 6.7 (third column of each cluster) by roles. Each table entry indicates the pair of roles that the pair of members involved in the gap played in the project, and the number of gaps identified for the respective pair of roles across the 5 sets of dependencies. The *Gap* column indicates the gaps identified by Cataldo, the *RG* column indicates the Real gaps identified by the Role-based measure, and the *FG* column indicates the False gaps which were also identified by the Role-based measure. Note that in the *RG* column I indicate with an "x" the pair of roles that should not coordinate directly according to the imposed organizational structure. In other words, in the *RG* column I indicate an "x" in a

Table 6.17: SHIP' STC index by dependency and by reason of communication

(a) Cataldo

Dep	RN	RC	CC	CA
D_1	0.09	0.33	0.33	0.28
D_2	0.08	0.50	0.39	0.37
D_3	0.10	0.39	0.40	0.33
D_4	0.10	0.34	0.37	0.31
D_5	0	0.53	0.37	0.32
Avg	0.07	0.42	0.37	0.32

(b) Role-based

Dep	RN	RC	CC	CA
D_1	0.10	0.30	0.38	0.31
D_2	0.12	0.52	0.52	0.52
D_3	0.12	0.37	0.45	0.39
D_4	0.12	0.33	0.43	0.37
D_5	0	0.46	0.46	0.46
Avg	0.09	0.40	0.45	0.41

table entry that maps the roles r_i and r_j when the position ij of the role-role matrix is equal 0 (refer to Chapter 4 for details). I show only the combination of pairs where there is at least one gap for any of the 3 types of gap.

In SHIP requirements are negotiated among business partners, the project managers, and the development leader (DevL). Therefore, no coordination during requirements negotiation is expected among any other role, or between other roles and these 3 involved with the project scope definition. The gap instances, then, reflect this organizational definition. Thus, it is not necessary to further analyze the gaps per type for this reason of communication (Table 6.17(c)).

For the three remaining reasons of communication, most of the gaps were identified as real gaps, indicating that most of the lack of coordination is between people who could have communicated if necessary. There was no imposed organizational structure stopping them of establishing contact with their colleagues. Despite the type of gap one may talk about, it is difficult to explain why certain communication did not take place because as a researcher I did not know up-front that a certain interaction was expected. Then, from the knowledge acquired about the project I supposed that the real gap instances did not take place because team members did not considered them needed. I have observed this team almost daily for entire three months, and I have seen team members standing up from their desks to ask colleagues about updates, to request help with technical issues or misunderstandings about requirements specifications, to brainstorm solutions together at any time despite of how busy they were with their own activities. Thus, I will rather focus on the discussion of backchannel communication instead of raising hypothesis about why members did not communicate. Backchannel communication represent interactions that in fact took place but that did not follow the organization structure. Thus, they

(c) Requirements negotiation

Roles	Gap	RG	FG
DevL-Dev	8	8	0
DevL-TestL	4	4	0
DevL-Tester	22	x	22
Dev-DevL	22	22	0
Dev-Dev	21	21	0
Dev-TestL	9	x	9
Dev-Tester	34	34	0
TestL-DevL	5	5	0
TestL-Dev	12	x	12
TestL-Tester	13	13	0
Tester-DevL	25	x	25
Tester-Dev	35	35	0
Tester-TestL	14	14	0
Tester-Tester	24	24	0
Total	248	180	68

(d) Requirements clarification

Roles	Gap	RG	FG
DevL-Dev	7	7	0
DevL-TestL	0	0	0
DevL-Tester	17	x	17
Dev-DevL	16	16	0
Dev-Dev	18	18	0
Dev-TestL	9	x	9
Dev-Tester	31	31	0
TestL-DevL	0	0	0
TestL-Dev	2	x	2
TestL-Tester	3	3	0
Tester-DevL	12	x	12
Tester-Dev	28	28	0
Tester-TestL	8	8	0
Tester-Tester	17	17	0
Total	168	128	40

(e) Communication of changes

Roles	Gap	RG	FG
DevL-Dev	3	3	0
DevL-TestL	0	0	0
DevL-Tester	17	x	17
Dev-DevL	4	4	0
Dev-Dev	15	15	0
Dev-TestL	6	x	6
Dev-Tester	31	31	0
TestL-DevL	0	0	0
TestL-Dev	10	x	10
TestL-Tester	0	0	0
Tester-DevL	25	x	25
Tester-Dev	35	35	0
Tester-TestL	5	5	0
Tester-Tester	21	21	0
Total	172	114	58

(f) Coordination of activities

Roles	Gap	RG	FG
DevL-Dev	5	5	0
DevL-TestL	4	4	0
DevL-Tester	22	x	22
Dev-DevL	7	7	0
Dev-Dev	15	15	0
Dev-TestL	7	x	7
Dev-Tester	31	31	0
TestL-DevL	8	8	0
TestL-Dev	9	x	9
TestL-Tester	0	0	0
Tester-DevL	25	x	25
Tester-Dev	35	35	0
Tester-TestL	8	8	0
Tester-Tester	11	11	0
Total	187	124	63

Table 6.18: SHIP's congruence gaps, real gaps, and false gaps

are understandable in an easier way than the congruence gaps in this context.

Actual communication by roles. I complete the in-depth analysis of requirements-driven coordination behavior and their alignment with coordination needs and with the organizational structure by exploring demographic data about the identified actual communication. I break down the actual communication information presented in Figure 6.7 (second column of each cluster) by roles. Similarly to the gap analysis, each table entry indicates the pair of roles that the pair of members involved in the actual communication played in the project, and the number of actual communication identified for the respective pair of roles across the 4 sets of dependencies. The *Ac* column indicates the actual communication identified by Cataldo, the *Al* column indicates the Aligned communication identified by the Role-based measure, and the *BCc* column indicates the Backchannel communication which is also identified by the Role-based measure. Note that in the *Al* column I indicate with an "x" the pair of roles that should not coordinate directly according to the imposed organizational structure. In other words, in the *Al* column I indicate an "x" in a table entry that maps the roles r_i and r_j when the position ij of the role-role matrix is equal 0 (refer to Chapter 4 for details).

Table 6.19 reveals that less than one-fifth of the actual communication across the four reasons of communication are backchannel communication according to the Role-based measure (55 out of 313). When one looks at the roles, developers (Dev) initiated about one-third (17 out of 55) of these backchannel communication, followed by the test leader (TestL) who alone initiated 15 of these "behind-the-scene" communication instances. However, when I group the roles by team a balance between development (27 out of 55) and test (28 out of 55) as initiators of backchannel communication is found. In addition, although I present results consolidated across the 5 sets of dependencies, three-quarters of the backchannel communication refers to 3 sets of dependencies that have a certain requirement in common. This requirements will be named $SHIP{-}R_5$ from now on. Table 6.19 also shows that the majority of the reported actual communication is considered aligned communication (258 out of 313). In other words, most of the communication took place between roles that indeed were expected to directly communicate with each other. Developer is the role who most initiated aligned communication (over one-third, 89 out of 258 instances).

More specifically, Table 6.19(a) shows few backchannel communication about requirements negotiation (over one-tenth, 3 out of 24). The 3 instances took place

Table 6.19: SHIP's actual, aligned, and backchannel communication

(a) Requirements negotiation

Roles	Ac	Al	BCc
DevL-Dev	9	9	0
DevL-DevL	5	5	0
DevL-TestL	4	4	0
DevL-Tester	0	x	0
Dev-DevL	0	0	0
Dev-Dev	0	0	0
Dev-TestL	3	x	3
Dev-Tester	0	0	0
TestL-DevL	3	3	0
TestL-Dev	0	x	0
TestL-Tester	0	0	0
Tester-DevL	0	x	0
Tester-Dev	0	0	0
Tester-TestL	0	0	0
Tester-Tester	0	0	0
Total	24	21	3

(b) Requirements clarification

Roles	Ac	Al	BCc
DevL-Dev	10	10	0
DevL-DevL	5	5	0
DevL-TestL	8	8	0
DevL-Tester	5	x	5
Dev-DevL	6	6	0
Dev-Dev	3	3	0
Dev-TestL	3	x	3
Dev-Tester	3	3	0
TestL-DevL	8	8	0
TestL-Dev	10	x	10
TestL-Tester	10	10	0
Tester-DevL	13	x	13
Tester-Dev	7	7	0
Tester-TestL	6	6	0
Tester-Tester	7	7	0
Total	104	73	31

(c) Communication of changes

Roles	Ac	Al	BCc
DevL-Dev	14	14	0
DevL-DevL	5	5	0
DevL-TestL	8	8	0
DevL-Tester	5	x	5
Dev-DevL	18	18	0
Dev-Dev	6	6	0
Dev-TestL	6	x	6
Dev-Tester	3	3	0
TestL-DevL	8	8	0
TestL-Dev	2	x	2
TestL-Tester	13	13	0
Tester-DevL	0	x	0
Tester-Dev	0	0	0
Tester-TestL	9	9	0
Tester-Tester	3	3	0
Total	100	87	13

(d) Coordination of activities

Roles	Ac	Al	BCc
DevL-Dev	12	12	0
DevL-DevL	5	5	0
DevL-TestL	4	4	0
DevL-Tester	0	x	0
Dev-DevL	15	15	0
Dev-Dev	6	6	0
Dev-TestL	5	x	5
Dev-Tester	3	3	0
TestL-DevL	0	0	0
TestL-Dev	3	x	3
TestL-Tester	13	13	0
Tester-DevL	0	x	0
Tester-Dev	0	0	0
Tester-TestL	6	6	0
Tester-Tester	13	13	0
Total	85	77	8

between the same developer in US and the test leader located in Brazil. The majority of the aligned communication (18 out of 21) were initiated by the development leaders.

A closer look in Table 6.19(b) reveals that three-quarters of backchannel communication (23 out of 31) about requirements clarification were initiated by the test team (either by a tester or by the test leader). When I look further the members' attributes I identified that most of these clarifications were requested to senior members of the development team. About one-third of the aligned communication (23 out of 73) was initiated by the development leaders.

In Table 6.19(c) see that the majority of the backchannel communication (11 out of 13) was initiated by the development team (either by a developer or by the development leaders). Also when I look further the members' attributes I identified that two-thirds (7 out of 11) of these change notifications were sent overseas from US to Brazil. The development team was also responsible for initiating about two-thirds of the aligned communication (54 out of 87). Each development role initiated 27 instances of aligned communication.

Less than one-tenth of the reported actual coordination of activities communication is considered backchannel communication (8 out of 85). In Table 6.19(d) see that these instances of backchannel communication took place between developers and the test leader (Dev-TestL equal 5, TestL-Dev equal 3). About one-third of the aligned communication (24 out of 77) was initiated by developers, and most of these instances (21 out of 24) are communication with development teammates.

6.2.5 Summary of Empirical Insights

In order to highlight the main insights gained in the SHIP case study, I summarize the insights by research question and measure in Table 6.20. Additional measures to the initial set of measures proposed in the framework are in bold.

Table 6.20: SHIP' summary of measures by requirements-driven collaboration aspect

RCSN Characterization		
Measure	*Question*	*Empirically-informed insight*
RCSN visualization	How does the RCSN look like?	Emergent and isolated members
RCSN size	How many people collaborating?	Over 1/3 are emergent
RCSN density	How tightly-coupled is the RCT?	1/3 of communication took place
Ties statistics	What kind of interactions?	High cross-sites and cross-teams
RCSN Structure		
Measure	*Question*	*Empirically-informed insight*
Centralization	Is collaboration centralized around certain members?	Decentralized collaboration
Core-periphery	Who is at the core of the RCSN?	Leaders and senior developers
Ties reciprocity	Do members collaborate in a equal manner with each other?	Mostly reciprocal communication
Clique	Who are the well-connected groups within the RCSN?	Leaders and senior developers
RCSN Information flow		
Measure	*Question*	*Empirically-informed insight*
Component	Are all members of the RCT connected?	There are disconnected members
Reachability	To what extent information can be shared with everyone?	Emergents can be reached but they cannot reach others
Cutpoints	Are there members that if removed would disrupt info flow?	Mainly the US dev leader
Degree	Who are the central members?	Mainly leaders
Brokerage	Who are the mediators of information?	Mainly the US dev leader
RCSNs Alignment		
Measure	*Question*	*Empirically-informed insight*
Alignment	To what extent are social relationships aligned to reqs dep? To what extent RCTs follow the organization structure?	Somehow aligned Mostly aligned

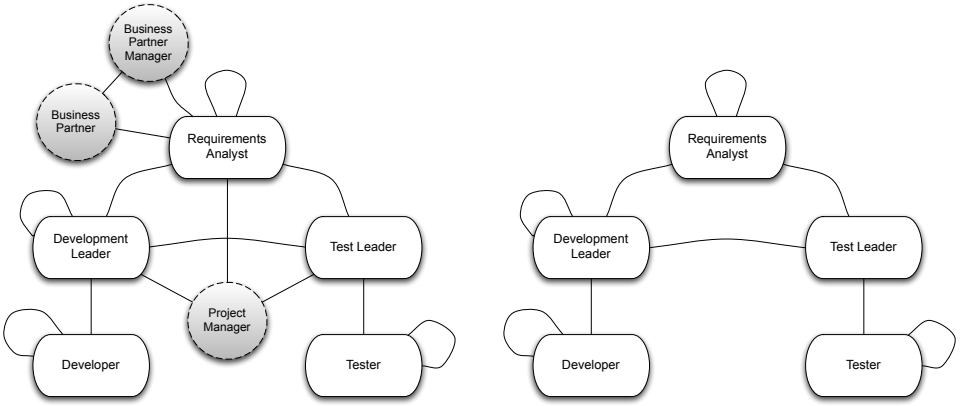
Chapter 7

Case 2. The Support Applications Project

In this chapter I present in details the examination of requirements-driven collaboration in the Support Applications project, shortened to APP. Similarly to the SHIP project, APP's project setting is first introduced followed by the examination of the requirements-centric social networks organized by research question. In addition, I discuss the threats to the validity of the empirical case study investigation of requirements-driven collaboration. These threats apply to both projects and are discussed here to facilitate the understanding of such threats.

7.1 Project Setting

Business area and project goal. APP, also a fictitious name, is a project that enhances and maintains a group of internal software products used by product management and sales. There are over one hundred mature software applications within this group, though about twenty are considered critical to the operation of the company. These applications are organized into four major groups by business area within APP's team. The project investigated was the first quarterly release of the support application project since the code ownership have been transfered to the Brazilian development unit. This means that the team was suffering demands from management to demonstrate ability to deliver on time and with the quality expected by the customers, who are ORG internal users.



(e) Reported organization structure (f) Reduced organization structure

Figure 7.1: APP's organization structure

Project requirements. In APP a list of desired improvements is maintained regularly. These desired improvements address business processes changes identified by the business partners. Each of the four main groups has its own business partner representative, who informs the requirements analysis about the desired improvements. There is also an additional business person who manages the business partners. By quarter, the improvements are prioritized and selected to comprise the quarter release scope. The business managers is the focal point to solve issues prioritizing the improvements. The improvements described in a "wish list" format are then translated into software requirements, and added to the existing product documentation. Each requirement is associated to one of the hundreds of applications that APP maintains. The wish list is formally approved in the end of the Envisioning phase, and it represents the project contract. Out of the 12-weeks long life-cycle of each release, three full weeks were dedicated to the analysis, prioritization, and specification of the requirements. Changes to the requirements are requested by the business partners, and impact analysis is conducted by the requirements analysts. In a management meeting, a representative of the business partners in addition to the project managers and requirements analysis decide together whether the changes are approved or added to the requirements back log for future implementation. Time to implement the change and value added to the application are two of the most often criteria used to decide on the acceptance of a change request.

Team distribution and office locations. APP's team consists of forty-four team members, distributed as follows: 38 in Brazil, 5 in US, and 1 in India. The 5 five

business partners are located in the US and 1 tester is located in India. The members located in Brazil are: 2 project managers, 4 requirements analysts, 1 test lead, 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 group. All team members are allocated to work full time on the project. About thirty percent are contractors or new hires in the company. Team members in Brazil are physically distributed in three buildings that are five minutes walking-distance of each other. The requirements analysts, development leaders, developers and the test leader work in the ORG's building, but in different offices. Two contractor developers work in the one contractor's building, and the test team works in the other contractor's building. Despite the fact that the buildings are physically close to each other, due to security policies team members have to identify themselves each time they visit the ORG or the contractors' buildings.

APP's organization structure, role and responsibilities. Figure 7.1(e) illustrates APP's organizational structure. The project managers supervise the overall work and report progress to the senior management. The business partners are responsible for identifying the desired improvements with their respective business teams, and to register these improvements in a "wish list" document. The requirements analysts analyze the whish list and discuss prioritization of improvements with the business partners and the project managers. The development leaders are invited to discuss the wish list when consensus about scope is not reached. They provide technical information that helps the group decide which improvement should be prioritized for a certain quarterly release. The test leader is also welcome. Once the scope is agreed, the requirements analysts translate the selected improvements into software requirements, which are formally documented. Note in Figure 7.1(e) that the requirements analysts intermediate communication between the project managers and the business partners.

The test team actively engages in the project after the improvements are translated into software requirements. The test leader is responsible for designing tools to automate test cases. Each tester is in charge of writing his own test cases, but he has to follow the automation strategy defined by the test leader. Developers are responsible for coding the requirements and performing integration tests of their code with other developers' code that are related to the same requirement. APP's organizational structure imposes that developers and testers cannot communicate directly

(see Figure 7.1(e)). They have to communicate with their leaders to reach someone in the other team, and the leader will discuss directly the matter with the target person.

Note that since the business partners, business manager, and the project managers do not have tasks allocated to the project, I did not consider them as part of the team. For the purpose of constructing the requirements-centric social networks, I considered only team members assigned to any task related to the project requirements. Thus, Figure 7.1(f) shows the reduced organization structure considered in this research.

Team expertise and communication practices. APP portfolio was transferred to Brazil six months before I have started the investigation. APP was formerly run by the headquarters office in the US. Most of the applications have no documentation about requirements, architecture, and how the applications depend on each other. The majority of the employees who had originally developed the applications have already left the company, thus there was low support from the headquarters to clarify questions about how the applications worked. To minimize the lack of expertise since every member in APP was new to the product, senior management decided to train APP's 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 four months conducting this process. Business partners in the US supported the team throughout this activity. Senior management also highlighted to the team members how important it was for them to follow the organizational structure to communicate and coordinate work. Formal weekly meetings and daily short meetings within and cross teams complement the senior management strategy to maintain awareness and control over the project work. E-mail is the main medium used by the Brazilian members to communicate in addition to face-to-face interaction. Contact with the business partners in the US is mostly mediated by group conference calls or one-to-one phone calls.

7.2 The Examination of Requirements-Driven Collaboration

Like in the SHIP case study, social network analysis measures were used as mechanisms to examine requirements-driven collaboration in the APP project. In practice, for each of the four sets of requirements dependencies in this project each measure

was applied to each communication- and awareness-requirements-centric social network previously constructed. To facilitate comprehension, the brief description of each measure will be repeated here. In this section I present the results of the application of these measures and interpret these results in terms of the project context.

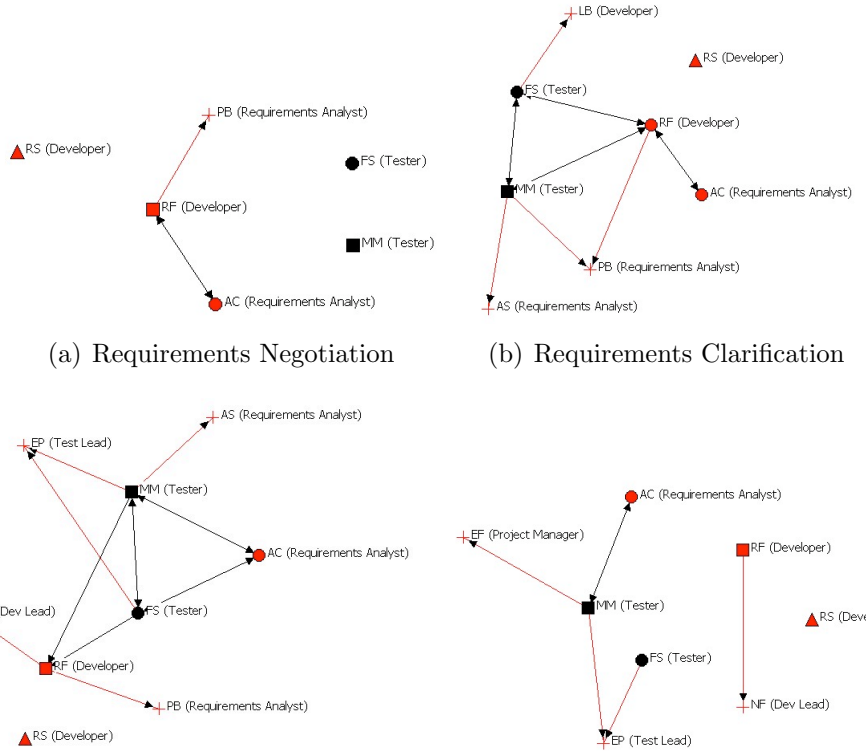
7.2.1 (RQ2) RCSNs Characterization

In the effort to characterize the communication- and the current awareness-RCSNs, I adopted the sociogram, network size, network density, and ties statistics measures defined in the initial version of the framework for studying requirements-driven collaboration. No measure was added to the framework to characterize the RCSNs like in the SHIP project.

RCSN sociogram. A sociogram is a picture in which actors are represented as points in two-dimensional space, and relationships among pairs of actors are represented by lines (or ties) linking the corresponding points. In addition to the actors and their relationships, sociograms can also indicate actors attributes or relationships characteristics. The same attributes presented in SHIP are displayed in APP, which are as follows. The member's fictitious name and role are indicated in the actor label, for example in Figure 7.2(a) the label *RF (Developer)* indicates a person named RF who is a developer in the APP project. The member's office location is indicated by the actor color (red for ORG's office and black for the contractor's office), for example the member *FS (Tester)* is located in the contractor's office. The requirement (or set of requirements) that the member was assigned to work on is represented by the actor shape (square for the dependent requirement, triangle for the dependee, circle for both requirements, and plus sign for emergent members who were not assigned to any requirement), for example *AC (Requirements Analyst)* was assigned to work on both requirements in the dependency set and *PB (Requirements Analyst)* is an emergent member. Whether the relationship is between members assigned to work on the requirements (black color) or with an emergent member (red color) is indicated by the color of the tie, for example the tie from *RF (Developer)* to *PB (Requirements Analyst)* represents an emergent communication.

As an illustration, Figure 7.2 displays the sociagram of each of the four reasons of communication for the dependency set D_1. One can see that the sociograms are different from each other but none is too dense or too sparse. Note that membership

varies because of the emergent members. One can also see in Figure 7.2(a) that there are isolated members, for example *RS (Developer)*. Interestingly, RS is isolated in each of RCSNs suggesting that somehow he did not participate in the development of the corresponding requirements.



(a) Requirements Negotiation (b) Requirements Clarification

(c) Communication of Changes (d) Coordination of Activities

Figure 7.2: APP's communication-RCSN sociograms

The sociogram for the current awareness-RCSN of the same dependency set D_1 is shown in Figure 7.3. Note that this network is not denser than any of the communication networks for this dependency set (the density measure confirms this result later on), meaning that team members are not more aware of their colleagues than they communicate with them. However, one can see that it has more emergent members than the communication networks, being the number of emergent people higher than of those assigned to work on the dependent requirements. This means that in this particular network, members are more aware of people who were not assigned to work in the requirements, raising the following questions: "Were these emergent members also working on these requirements?" "Why was the tester *MM* aware of the emergent members' work if they were not working in these requirements?". Moreover, note

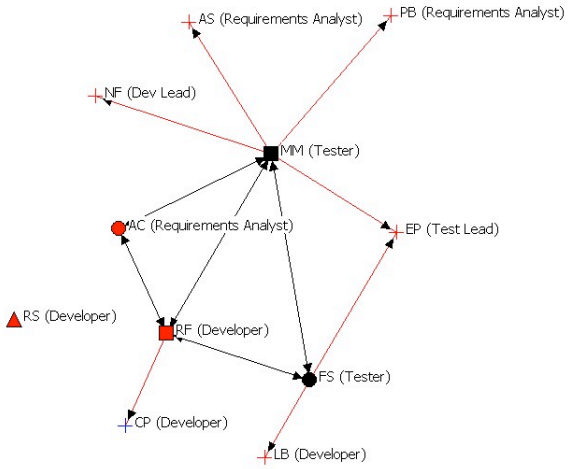


Figure 7.3: APP's current awareness-RCSN sociograms

that the tester *MM* is responsible for the involvement of two-thirds of the emergent members in this network.

Since the sociograms were used to develop an initial broad understanding of each network behavior and to explore patterns of requirements-driven collaboration later examined in details using the social network analysis measures, the remaining sociograms in the data set are not displayed in the body of this document. Refer to Appendix C for the presentation of each of the communication- and awareness-RCSN sociograms.

RCSN size. Each RCSN constructed for this project was initially based on a baseline network called assignment-RCSN, as defined in the framework (see Chapter 4). Each of these assignment network consists of a requirements-centric team which is composed of a subset of the 11 team members who were assigned to the project scope investigated in this research. I refer to these members as *assigned members* from now on. To construct the corresponding communication-RCSN, I identified from the questionnaire data which project members listed from the assignment-RCSN that the respondents identified as communicating about the requirement and any other member that the respondents mentioned as additional people they have communicated with about the requirement. I name these additions *emergent members* from here onwards. They complement the requirements-centric teams. Therefore, each RCSN has a certain number of members, named *actual members*, which are divided into assigned and emergent members.

Table 7.1: APP's distribution of assigned members per communication-RCSNs by location

	Assigned	
	BR	US
D_1	3	2
D_2	2	2
D_3	3	2
D_4	3	2
$Mean$	2.8	2

Table 7.1 shows the distribution of assigned members by office location (ORG for the ORG's main office, and C1 for the contractor company 1) for any of the reasons of communication per dependency set, and Table 7.2(a) shows the distribution of actual and emergent members for the corresponding Requirements negotiation networks. Actual members are variable because the number of emergent members varies per network. A total of 4 emergent members across the 4 dependency sets were identified, all of them located in the ORG office. These are non-unique people, i.e. there are duplicates across RCSNs. On average, there are a total of 1 emergent people per network. This indicates that, on average, less than 20% of the requirements-centric team members out of the total are emergent in a RCSN that communicated about requirements negotiation.

Similarly, Table 7.2(b) shows the distribution of actual and emergent members for the corresponding Requirements clarification networks. A total of 10 non-unique emergent members was found. On average, there are a total of 2.5 emergent people per network. This indicates that, on average, about 35% of the members clarifying requirements are emergent to the requirements dependency set. The distribution of actual and emergent members for the Communication of changes networks is presented in Table 7.2(c). There are 13 non-unique emergent members across these networks. On average, there are a total of 3.3 emergent people per network. In the Requirements clarification networks the emergent members represent about 40% of all members who discussed changes in the project. Table 7.2(d) presents the distribution for the Coordination of activities networks. There are 10 non-unique emergent members across these networks. Each network has, on average, 4.4 emergent members. These emergent members represent about 35% of those involved with coordinating activities. For the four reasons of communication, members located at the ORG office are among the only emergent members.

Table 7.2: APP's distribution of members per communication-RCSNs by location

(a) Requirements negotiation

	Actual		Emergent	
	ORG	C1	ORG	C1
D_1	4	2	1	0
D_2	3	2	1	0
D_3	4	2	1	0
D_4	4	2	1	0
$Mean$	3.8	2	1	0

(b) Requirements clarification

	Actual		Emergent	
	ORG	C1	ORG	C1
D_1	6	2	3	0
D_2	5	2	3	0
D_3	6	2	3	0
D_4	4	2	1	0
$Mean$	5.3	2	2.5	0

(c) Communication of changes

	Actual		Emergent	
	ORG	C1	ORG	C1
D_1	7	2	4	0
D_2	6	2	4	0
D_3	7	2	4	0
D_4	4	2	1	0
$Mean$	6	2	3.3	0

(d) Coordination of activities

	Actual		Emergent	
	ORG	C1	ORG	C1
D_1	6	2	3	0
D_2	5	2	3	0
D_3	6	2	3	0
D_4	4	2	1	0
$Mean$	5.3	2	2.5	0

Table 7.3: APP's distribution of members per awareness-RCSNs by location

	Assigned			Actual			Emergent		
	ORG	C1	C2	ORG	C1	C2	ORG	C1	C2
D_1	3	2	0	8	2	1	5	0	1
D_2	2	2	0	7	2	1	5	0	1
D_3	3	2	0	8	2	1	5	0	1
D_4	3	2	0	4	2	0	1	0	0
$Mean$	2.8	2	0	6.8	2	0.8	4	0	0.8

The distribution of team members per awareness-RCSN per set of dependency by location is showed in Table 7.3. Note that members are also aware of colleagues who work for the second contractor company, named C2. A total of 19 non-unique emergent members across the 4 dependency sets were identified, 85% of them (16 members) are located in the ORG office. These are non-unique people, i.e. there are duplicates across RCSNs. On average, there are a total of 4.8 emergent people per network. This reveals that, on average, 50% of the members that team members are aware of are emergent, suggesting that collocation facilitated the ability of the APP's team members of staying aware of their colleagues.

Table 7.4: APP's communication-RCSNs density

(a) With emergent members

Dep	RN	RC	CC	CA
D_1	0.10	0.28	0.18	0.10
D_2	0.15	0.28	0.23	0.14
D_3	0.10	0.21	0.18	0.11
D_4	0.26	0.36	0.43	0.06
$Mean$	0.15	0.28	0.25	0.10

(b) Without emergent members

Dep	RN	RC	CC	CA
D_1	0.10	0.40	0.40	0.10
D_2	0.16	0.66	0.66	0.16
D_3	0.10	0.40	0.40	0.10
D_4	0.35	0.45	0.50	0
$Mean$	0.17	0.47	0.49	0.09

RCSN density. Network density is defined as the proportion of ties that exist in the network out of the total possible ties. Table 7.4(a) shows the density for the communication-RCSNs. One can see that the densities are quite low, specially the Coordination of activities networks. These low indexes indicate that the team did not discuss much project issues in a one-to-one manner. During my visit on site I observed weekly meetings within developers and daily short meetings where people would report issues and describe the day activities to the colleagues among testers. The occurrence of these weekly and daily meetings may have diminished the need for one-to-one conversations. To exclude the hypothesis that the emergent members are responsible for the low network densities (for only being consulted by assigned members and not initiating communication themselves), I removed them and recalculated the density for each network. Table 7.4(b) shows that most of the networks have its density value increased. However, Requirements negotiation and Coordination of activities maintained their low indexes. Thus, one can conclude that team members did not interact much about these two reasons of communication.

Awareness-RCSNs also have low indexes (refer to Table 7.5(a)), suggesting that team members were not as aware of many colleagues as one would expect since they were working on dependent requirements. However, when removing the emergent members similarly like for the communication networks, one can see in Table 7.5(b) that the indexes are twice higher on average. This indicates that in fact the assigned members were aware of at least 50% of their requirements-centric team members for each network. Considering that the APP team was new and most of the members have never worked together, this low lack of awareness is not surprising. Apparently, it was also not prejudicial since no dramatic episodes caused for lack of up-to-dated information related to the requirements that one was working on or knowledge of others was observed.

Table 7.5: APP's current awareness-RCSNs density

(a) With emergent members

Dep	Awareness
D_1	0.15
D_2	0.20
D_3	0.16
D_4	0.53
$Mean$	0.26

(b) Without emergent members

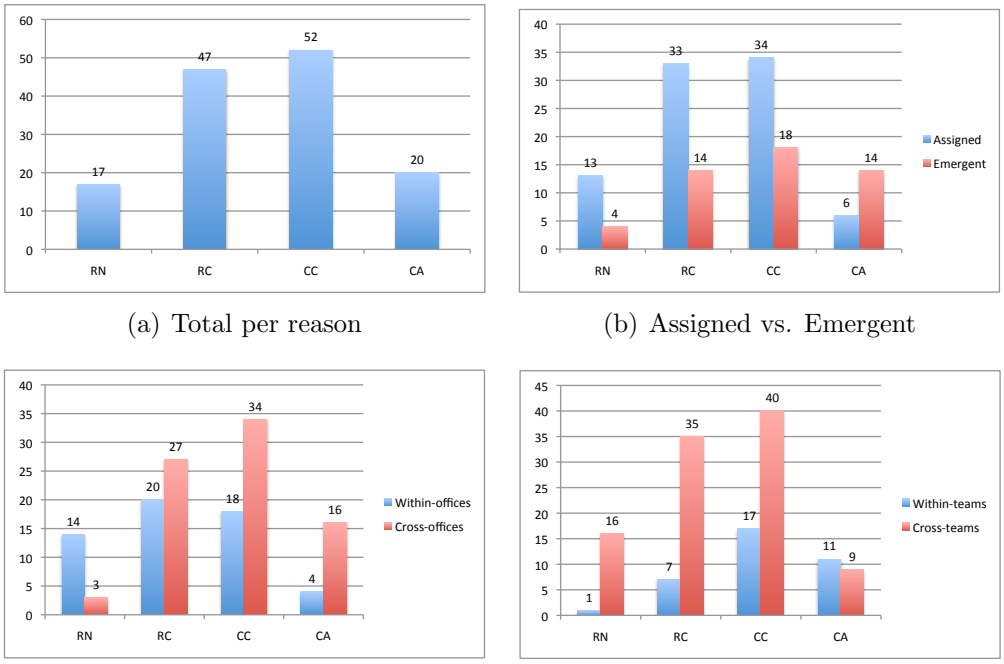
Dep	Awareness
D_1	0.50
D_2	0.83
D_3	0.50
D_4	0.65
$Mean$	0.62

Ties statistics based on team members attributes. To identify patterns of collaboration and information exchange within RCSNs, I provide descriptive statistics about the relationships established by the members based on their attributes. For both the communication- and the awareness-RCSNs, I counted the ties by requirement dependency set in terms of which members are involved (assigned or emergent members), what is the location of the members (within- or cross-offices), and team membership (within- or cross-teams).

Communication-RCSNs. Figure 7.4 shows the consolidated results across the 4 sets of dependencies per reason of communication for simplicity. Appendix C presents the ties statistics per individual set of dependency.

In Figure 7.4(a) one can see that a total of 17 communication interactions about requirements negotiation (RN) were reported, as well as 47 interactions about requirements clarification (RC), 52 about communication of changes (CC), and 20 about coordination of activities (CA) were reported by the respondents, summing up a total of 136 interactions. From this figure one see that communication of changes is the most frequent topic reported, followed by requirements clarification.

Figure 7.4(b) presents the number of times per reason of communication an reported interaction took place between members who were assigned to work on the requirements (assigned) and with an emergent member (emergent). A total of 86 (out of 136, 63.24%) interactions were reported between assigned members and 50 (36.76%) with emergent members. This indicates that about one-third of the development work involves communication with people who were not assigned to work on the requirements, suggesting that project members seek a large amount of information from outsiders. From Figure 7.4(b), one see that the requirements clarification and communication of changes are the two most-often reported reasons of interaction



(a) Total per reason

(b) Assigned vs. Emergent

(c) Within- vs. Cross-offices

(d) Within- vs. Cross-teams

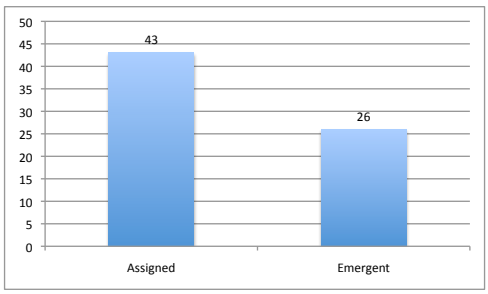
Figure 7.4: APP's communication-RCSN across dependency sets ties statistics

between assigned members, and that no single reason stands out among the emergent interactions.

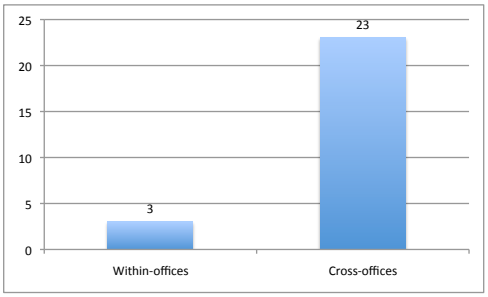
Figure 7.4(c) presents communication per reason characterized by office location. Within-offices indicates communication between two members at the same office, and across-offices indicates communication between two members at different offices. A total of 56 (out of 136, 41.18%) interactions were reported between members in the same office and 80 (58.82%) between members at different offices. Team members communicated more with members from the contractors companies, who are the members located in different offices, than with collocated colleagues. Considering that there were more members assigned to the ORG office, this difference is even more significant and meaningful. Looking at the data in Figure 7.4(c), one see that communication of changes (34/80) is the most frequent reason why members communicated with colleagues from other offices.

When I sorted the instances of communication between members belonging to the same team (within-teams) and between members belonging to different teams (cross-teams), the networks are mostly characterized by cross-teams communication (see Figure 7.4(d)). Three-quarters of the reported communication took place cross-

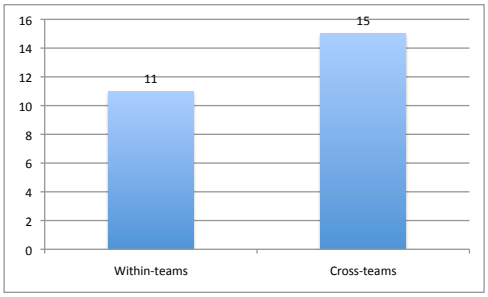
teams. This high percentage of cross-teams communication suggests that the teams are well integrated and collaborate with each other to accomplish the project goals. No pattern stands out across the within-teams communication reasons, however one can see that requirements clarification and communication of changes are the two most-often reported reasons of communication cross-teams (refer to Figure 7.4(d)). Not only the team members communicated more with remote colleagues but also with people outside their teams.



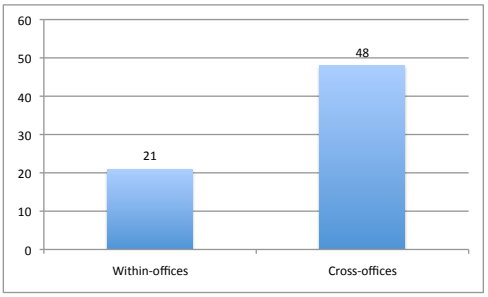
(a) Assigned vs. Emergent



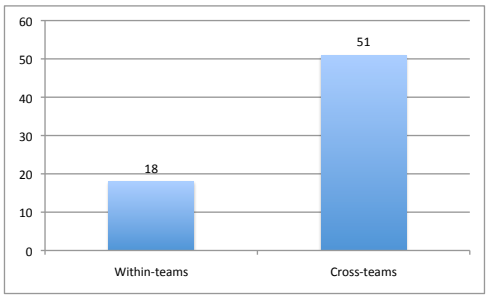
(b) Emergent within- vs. cross-offices



(c) Emergent within- vs. cross-teams



(d) Within- vs. Cross-offices



(e) Within- vs. Cross-teams

Figure 7.5: APP's current awareness-RCSN across dependency sets ties statistics

Current awareness-RCSNs. Figure 7.5 shows the consolidated results for the current awareness among team members across the 4 sets of dependencies for simplicity.

Appendix C presents the ties statistics per individual set of dependency. A total of 69 instances of awareness were reported. Of these, in Figure 7.5(a) one see that 43 instances (62.32%) represent awareness between members assigned to work on the requirements (assigned) and 26 instances (37.68%) represent awareness of emergent members (emergent). This indicates that not only more than half of all members of the requirements-centric teams are aware of their colleagues but also that this awareness is extended for those who emerged throughout the project life-cycle.

Of the 26 instances of reported awareness of emergent members, in Figure 7.5(b) one can see that only 3 instances (11.54%) represent awareness of members working on the same site and 23 represent awareness of members located in other offices (88.46%). This indicates that awareness of remote members are more likely to emerge. Although members are located in three distinct building that are about 10 minutes away walking from each other, there is a strict security protocol to enter the buildings which often stopped people from meeting face-to-face. In addition, Figure 7.5(c) shows that 11 instances (42.31%) represent awareness of teammates and 15 represent awareness of members working in other teams (57.69%). This indicates that awareness of teammates is almost as likely to emergent as of members of other teams.

Figure 7.5(d) shows that awareness of collocated colleagues (43 out of 69, 62.32%) is almost half higher than awareness of remote members (26 out of 69, 37.68%). This reveals that team members are more knowledgeable of what people working in the same office are doing than with what remote colleagues are working on. This suggests that distance, despite short, may impact the ability of team members of keeping track of remote people's work. Figure 7.5(e) also shows that team members are less aware of what their teammates are doing (18 out of 69, 26.09%) than what people in other teams are working on (51 out of 69, 73.91%). This result suggests that there is large exchange of information among team members about what colleagues outside one's team are working on.

7.2.2 (RQ3) RCSNs Structures

Like in the SHIP project, in order to identify what structures communication- and current awareness-RCSNs have in requirements-driven collaboration, I used the network centralization measure defined in the initial version of the proposed framework. Supplementing this single measure, I used the measures in the SHIP project, which are: the *core-periphery* test, the *ties reciprocity* measure, and the *clique* measure.

RCSN centralization. A centralized network structure (index towards value 1) will have many of its ties dispersed around one or a few nodes, while a decentralized network structure (index towards value 0) is one in which there is little variation between the number of ties each node possesses. Table 7.6(a) shows the degree of centralization of communication ties sent to others, called outdegree centralization. There is no strong pattern across the reasons of communication or the sets of dependencies. However, no single index is higher than 0.55 (CC_{D2}), and 11 out of the 16 networks have an index lower than 0.50. This trend suggest that the networks are more decentralized than centralized in few members, meaning that in APP the distribution of information is more equally distributed among the team members than controlled by few of them.

The indegree index, presented in Table 7.6(b), refer to the degree of centralization of communication ties received from others. All networks have low indexes, varying from 0.06 to 0.25, except for RC_{D4} (0.52) and CC_{D4} (0.44). Requirements negotiation and Communication of changes have indexes particularly lower than the Requirements clarification and the Coordination of activities reasons. These low indexes indicate that APP has very decentralized indegree network structures. One can then affirm that information is more equally received by all team members within the requirements-centric social networks.

Table 7.7(a) shows the outdegree centralization index for the awareness-RCSNS. One can see that indexes are higher than 0.59, an exception is the $Awareness_{D4}$ network. These high indexes indicate that few people are aware of many of their colleagues. In contrast, the indegree centralization index for the awareness-RCSNs presented in Table 7.7(b), which indicates the extent to what the work of colleagues in the network is known by others, shows low indexes. This result suggests that team members are equally known by others in the project.

Table 7.6: APP's communication-RCSN centralization

(a) Outdegree

Dep	RN	RC	CC	CA
D_1	0.36	0.44	0.50	0.37
D_2	0.44	0.44	0.55	0.41
D_3	0.36	0.41	0.50	0.37
D_4	0.40	0.52	0.44	0.16
$Mean$	0.39	0.45	0.50	0.33

(b) Indegree

Dep	RN	RC	CC	CA
D_1	0.12	0.25	0.08	0.20
D_2	0.12	0.25	0.06	0.22
D_3	0.12	0.24	0.08	0.20
D_4	0.40	0.52	0.44	0.40
$Mean$	0.19	0.32	0.16	25.7

Table 7.7: APP's current awareness-RCSN centralization

(a) Outdegree

Dep	Awareness
D_1	0.60
D_2	0.64
D_3	0.59
D_4	0.32
$Mean$	0.54

(b) Indegree

Dep	Awareness (%)
D_1	0.16
D_2	0.15
D_3	0.15
D_4	0.32
$Mean$	0.19

Core-periphery. The core-periphery test indicates the extent to which the structure of a network consists of two classes of nodes: the core, in which nodes are connected to each other in some maximal sense, and the periphery, where nodes are more loosely connected to the core. A value close to 1 indicates a strong core-periphery structure. Table 7.8(a) shows the values for the core-periphery test for the communication-RCSNs. Like presented in the SHIP project, this is an additional measure to the framework.

Most of the networks have high indexes that range from 0.68 to 1. These high indexes indicate that a structure of a core and a surrounding periphery is predominant in the APP communication requirements-centric social networks. This finding somehow contradicts the network centralization results where it was found more decentralized communication structures. It may be explained by the high presence of emergent members and isolated members in each network. When the list of core members provided by the test was inspected (see Appendix C for the complete list), I found that most of network cores are formed by a requirements analyst and testers. Surprisingly developers, who because of the reverse engineering activity were the most knowledgeable members of the team, are not involved in the core as much as expected. On the other hand, I did had the chance to observe the testers closely working together with their teammates and having informal meetings with requirements analysts to discuss the requirements specifications.

Ties reciprocity. Another additional measure to the framework is the ties reciprocity measure. In a directed network, one can examine the percentage of pairs of actors that are involved in a reciprocal relationship (dyad-based method) or the percentage of ties that have a reciprocated tie (arc-based method). Table 7.9(a) and Table 7.9(b) present the reciprocity percentages for each method, respectively. For

Table 7.8: APP's RCSN core-periphery index

(a) Communication

Dep	RN	RC	CC	CA
D_1	1	1	0.74	0.68
D_2	1	1	0.71	0.67
D_3	1	1	0.74	0.68
D_4	0.35	0	0	0.68
$Mean$	0.84	0.75	0.55	0.68

(b) Current awareness

Dep	Awareness
D_1	0.89
D_2	0.88
D_3	0.89
D_4	0.28
$Mean$	0.73

both methods the percentages are balanced between higher for the Requirements negotiation and Requirements clarification networks and lower for the Communication of changes and the Coordination of activities networks. I visually inspected the networks aiming to grasp whether reciprocity have been affected by the presence of the emergent members, and I noticed that the percentages would still be low if emergent members had been removed. Thus, overall, results suggest that communication is not highly mutually exchanged in the requirements-centric social networks. A more fine-grained analysis of which pair of members are involved in reciprocal communication would reveal whether communication is one-way because of the role that the members play in the project (e.g., a development leader may be expected to communicate changes to developers but not necessarily developers have to return the notification).

Table 7.9: APP's communication-RCSN ties reciprocity

(a) Dyad-based method

Dep	RN	RC	CC	CA
D_1	0.50	0.50	0.30	0.20
D_2	0.50	0.50	0.30	0.20
D_3	0.50	0.50	0.30	0.20
D_4	06.0	0.57	0.44	0
$Mean$	0.52	0.51	0.33	0.15

(b) Arc-based method

Dep	RN	RC	CC	CA
D_1	0.66	0.66	0.46	0.33
D_2	0.66	0.66	0.46	0.33
D_3	0.66	0.66	0.46	0.33
D_4	0.75	0.73	0.61	0
$Mean$	0.68	0.67	0.50	0.25

For the awareness-RCSNs, the arc-base method percentages are higher than the dyad-based method, indicating that there are more awareness reciprocity between members than pairs of team members that are aware of each other. However, the graphical inspection of the awareness networks reveals that if the emergent members were to be removed, about every single pair of assigned member is aware of each other, also indicating that every awareness relationship has a reciprocal. This visual

Table 7.10: APP's awareness-RCSN ties reciprocity

(a) Dyad-based method

Dep	Awareness
D_1	0.42
D_2	0.38
D_3	0.38
D_4	0.45
$Mean$	0.41

(b) Arc-based method

Dep	Awareness
D_1	0.59
D_2	0.55
D_3	0.55
D_4	0.62
$Mean$	0.58

inspection reveals that the ties reciprocity indexes would be 1 for each network if the emergent members were removed. In fact, this result indicate that despite the low levels of communication among requirements-centric team members and the team be working together for a short period of time, members are reciprocally aware of what others are doing that is related to their work. This is a surprising result due to the team unfamiliarity with each other and with the product.

Clique. A clique consists of a subset of at least three actors of a network in which every possible pair of actors is directly connected by a tie and this clique is not contained in any other clique. In directed networks there are two types of cliques: the strong, where only reciprocal ties between pairs of actors are considered, and the weak, when the direction of the tie is disregarded and simply the presence or absence of a relation is considered. This measure is an addition to the proposed framework.

Since no strong clique could be computed for most of the networks because emergent members did not respond about their communication interactions, and thus there is no reciprocal relationships for most of the interactions with these members, Table 7.11 presents the number of weak cliques found for the communication-RCSNs. No pattern across the reasons of communication or the set of dependencies stands out. However, one can see that Requirements negotiation and the Coordination of activities networks have no cliques, meaning that no subgroups with completely connected members were formed. In other words, there is no team members discussing project issues closely to other teammates. Interestingly, the few cliques formed in the Requirements clarification networks are among three roles: a requirements analyst, a developer, and a tester. This result indicates a representative of each subteam in the project is fully connected to the other team representatives, suggesting that no group was left behind when it came to have discussions aiming to understand the project

Table 7.11: APP's communication-RCSN weak cliques

Dep	RN	RC	CC	CA
D_1	0	2	3	0
D_2	0	2	3	0
D_3	0	2	3	0
D_4	0	2	3	0

requirements. In the Communication of changes networks, most of the cliques few identified cliques are formed by a developer and two testers, suggesting that developers propagated the notification of changes ahead to their tester colleagues.

7.2.3 (RQ4) RCSNs Information Flow

To identify specific patterns of information exchange and knowledge sharing, I applied the component, and the degree centrality measures defined in the proposed framework. Like in the SHIP project, I added to other measures to this initial set, which are reachability and cutpoint.

Component. The component test indicates whether a social network is connected. The network is connected if there is a path between every pair of nodes, otherwise it is disconnected. The connected subsets in a social network are called components. For directed networks, two kinds of components can be defined: a weak component, where a set of actors are connected regardless of the direction of ties, and a strong component, where a directed path from actor A to actor B is expected for the two actors be in the same component.

Table 7.12(a) shows the amount of components found when the direction of the ties are ignored. Note that only 4 out of the 16 networks are formed by one component only. This indicates that most of the networks have isolated members who were assigned to work on the requirements. A more in-depth analysis by inspecting the network sociograms (refer to Appendix C) reveals that testers are the isolated people in the Requirements negotiation networks. This isolation corroborates the role description of testers, who are not supposed to get involved with the definition of the project scope. For both the Requirements clarification and Communication of changes the only isolated member is a new hired contractor developer named *RS*. RS has just been hired by a contractor company that provides development service to ORG, and was still finishing his on boarding training at ORG. This is likely the reason why he

Table 7.12: APP's communication-RCSN component

(a) Weak component

Dep	RN	RC	CC	CA
D_1	4	2	2	3
D_2	3	1	1	2
D_3	4	2	2	3
D_4	1	1	1	4

(b) Strong component

Dep	RN	RC	CC	CA
D_1	5	5	7	7
D_2	4	4	6	6
D_3	5	5	7	7
D_4	3	2	2	6

did not report having communicated with his colleagues. Interestingly, the pattern found in the Coordination of activities networks reveals two groups, one formed by another among developers, and another by a project manager, a requirements analyst, and testers. The second group (component) shows that testers used the requirements analysts and the project managers as source of information more than they sought developers for discussing project matters. If one consider the direction of the ties (see Table 7.12(b)), the networks have even more components than in the weak component structure. This suggests that the disconnection in the requirements-centric teams members goes beyond the isolated members.

Reachability. To better characterize patterns of information exchange and knowledge sharing, the reachability measure was also adopted in the framework. This measure denes that an actor is reachable by another actor if exists any set of ties that connects both actors, regardless of how many others fall between them. In a directed network directed paths between actors are considered. I have used this measure as a resource to compliment the results of the component measure. Due to the large number of communication networks investigated in this research, the resulting reachability matrix for each network are presented in the Appendix C. Here I highlight the main patterns found. Figure 7.6 shows the reachability matrix for the Requirements negotiation-RCSN for the dependency set D_1 as an example. In this network, particularly, most of the members (4 out of 6) cannot reach each other. The network sociogram shows that out of these 4 members, 3 are assigned members who are isolated from the remaining of the network, and 1 is an emergent member who is not expected to initiate communication with other since he was not assigned to work on the requirements.

The overall patterns identified when initially inspecting the sociograms to develop a broad idea of the networks behavior is confirmed by this measure, which are as

	R	A	F	M	R	P
RF (Developer)	0	1	0	0	0	1
AC (Requirements Analyst)	1	0	0	0	0	1
FS (Tester)	0	0	0	0	0	0
MM (Tester)	0	0	0	0	0	0
RS (Developer)	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0

Figure 7.6: APP's Requirements Negotiation-RCSN reachability matrix for dependency set D_1

follows. In the Requirements negotiation networks, the test members cannot reach or be reached by colleagues, indicating that they are isolates. In the Requirements clarification and Communication of changes network people are better connected, everyone can reach everyone else except the emergent members who cannot reach the other members. A single developer and a single tester are isolated members across the 4 Coordination of activities-RCSNs.

Cutpoint (or cutsets). Another additional measure is the cutpoint measure. A cutpoint is an actor identified as a weak point in the network because if the actor is removed along with his ties, the network would become divided into unconnected parts. If there is a set of actors that fit this criteria, these actors are called a cutset. Table 7.13 presents the list of cutpoints (or cutsets) for each communication-RCSN. Note that most of the networks have a cutset, indicating that it is necessary to remove more than one member to disrupt the exchange of information among the requirements-centric team members. Although from the list one find that members of the test team are predominant cutpoints within the cutsets, when inspecting the sociograms one can see that the cutsets are composed of the assigned members except the requirements analysts. Since the APP networks are small, in fact the cutsets represent the core and the most active members. This finding is corroborated by the results of the core-periphery test and of the degree centrality measure.

Degree centrality. The degree centrality indicates the number of ties of an actor and is indicative of activity. In a directed network, the count is made for out-ties (outdegree) which are ties from a certain actor to others, and for in-ties (indegree) which are ties from others to a certain actor. A complete list of the degree centrality for each set of requirements dependency is presented in the Appendix C. Table 7.14

Table 7.13: APP's communication-RCSN cutpoints

Dep	RN	RC	CC	CA
D_1	RF (Dev)	RF (Dev) FS (Tester) MM (Tester)	RF (Dev) MM (Tester)	MM (Tester) EP (Test Lead)
D_2	RF (Dev)	RF (Dev) FS (Tester) MM (Tester)	RF (Dev) MM (Tester)	MM (Tester) EP (Test Lead)
D_3	RF (Dev)	RF (Dev) MM (Tester)	RF (Dev) MM (Tester)	MM (Tester) EP (Test Lead)
D_4	AS (Req An) PF (Dev)	AS (Req An)	– none –	WR (Dev Lead)

highlights the team member (or set of members in case that the same degree was found for more than one person within a RCSN, indicated with a *) who have the highest communication degree centrality by dependency set.

Table 7.14(a) presents who most communicated with others. Note that like in the cutpoint measure, here testers are the most predominant members across the 4 reasons of communication. This indicates that testers the members who most initiated conversations with colleagues, except for the Requirements negotiation networks where the developer named *RF* shows as the most active member.

No pattern stands out in the indegree measure when observing the most active members across all networks, except that the requirements analysts are among the ones who most received information from their colleagues. However, there are interesting patterns across the dependency sets for each reason of communication. The most interesting pattern is the one that reveals that developers were the members most sought by colleagues to clarify requirements. This finding would be surprising but in this project the development team spent three months conducting reverse engineering in the applications aiming to understand and document the requirements. Thus, it is natural that they became the focal points for discussion. One can also see that the testers are the ones who most received notification of changes, which suggests that the APP team maintained the test team informed of updates.

Table 7.15(a) shows the member (or set of members, indicated with a *) who is more aware of others for each dependency set. Surprisingly, a contractor tester named *MM* is the member most aware of this requirements-centric team colleagues. He is also the person who the colleagues are most aware of, however he shares the attention

Table 7.14: APP's communication-RCSN highest degree centrality member

(a) Outdegree

Dep	RN	RC	CC	CA
D_1	RF (Dev)	Dev or Tester	MM (Tester)	MM (Tester)
D_2	RF (Dev)	Dev or Tester	MM (Tester)	MM (Tester)
D_3	RF (Dev)	Dev or Tester	MM (Tester)	MM (Tester)
D_4	ReqAn or Dev*	AS (ReqAn)	AS (ReqAn)	Dev*

(b) Indegree

Dep	RN	RC	CC	CA
D_1	ReqAn, Dev*	RF (Dev)	ReqAn, Test team*	EP (TestL)
D_2	ReqAn, Dev*	RF (Dev)	ReqAn, Test team*	EP (TestL)
D_3	ReqAn, Dev*	RF (Dev)	ReqAn, Test team*	EP (TestL)
D_4	AS (ReqAn)	AS (ReqAn)	AS (ReqAn)	WR (DevL)

Table 7.15: APP's awareness-RCSN highest degree centrality member

(a) Outdegree

Dep	Awareness
D_1	MM (Tester)
D_2	MM (Tester)
D_3	MM (Tester)
D_4	ReqAn, Dev*

(b) Indegree

Dep	Awareness
D_1	Dev, Tester*
D_2	Dev, Tester*
D_3	Dev, Tester*
D_4	AS (ReqAn)

from others with a contractor developer named *RF*.

7.2.4 (RQ5) RCSNs Alignment

To examined the alignment between RCSNs I used the current socio-technical congruence measure and the Role-based measure as presented in the initial version of the framework. No additional measures were defined.

Alignment of networks. Similarly than in the SHIP case study, in order to examine to what extent requirements-driven collaboration patterns are aligned with coordination needs established by requirements dependencies, I used the socio-technical congruence measure proposed by Cataldo et al. [24]. This measure calculates the ratio of actual social interactions over the expected coordination needs defined based on the technical dependencies in the project (refer to Chapter 2 for details).

To investigate to what extent requirements-driven collaboration patterns are aligned

with the organizational structure defined by the communication channels established between pairs of roles I used the *Role-based socio-technical congruence* measure presented in Chapter 4. The application of this extended measure allowed the identification of false gaps and backchannel communication. Next I present the results found by both measures, constantly contrasting one against the other.

Calculating Cataldo's congruence. For each measure, I calculated coordination needs and actual coordination according to the definitions presented in Chapter 2 and Chapter 4. I will refer here to each definition and equation presented in these chapters for simplicity like previously explained in the SHIP project. To calculate Cataldo congruence index (Equation 2.2), I computed the coordination needs matrix followed by the actual coordination matrix. To compute the *coordination needs matrix* (Equation 2.1), I constructed the *tasks assignment* and *tasks dependency matrices* from the requirements dependencies and team members data gathered. Since my unit of analysis is a requirement, I will rename the "tasks assignment" matrix and "tasks dependency" matrix to *requirements assignment* matrix and *requirements dependency* matrix, respectively. I computed 16 *actual communication matrices* for APP from the reported questionnaire communication interactions data. Each matrix represent coordination activities that took place for a certain set of requirements dependency and were about a certain reason. Remember that there are 4 sets of dependencies for APP, and that I asked team members to report on 4 reasons of communication. For example, the APP $D_1 com_{RN}$ actual coordination network captures requirements negotiation communication among team members for the set of requirements that belong to the dependency D_1. I calculated a *lack-of-coordination matrix* (Equation 2.3) to identify the congruence gaps by each combination of each set of dependencies and each reason of communication. For each gap instance, I gathered demographic information on the pair of members involved on the gap. In addition to the team member role, demographic information include the team member office location, time working in the project, time working in the company, and the work relationship (employee or contractor).

Calculating the Role-based congruence. As presented in Chapter 4, to calculate the Role-based congruence index (Equation 4.4), I computed the prime coordination needs matrix followed by the alignment coordination matrix. To compute the *prime coordination needs matrix* (Equation 4.2), I used the *requirements assignment*

matrix and *requirements dependency matrix* previously constructed and constructed two other matrices. I constructed the *role-role matrix* from the organization' structure data and the *role-assignment matrix* as described in Chapter 4. Then, I calculated the *role-coordination* matrix (Equation). To compute the *alignment coordination matrix* (Equation 4.3), I used the *actual communication matrices* previously constructed and the prime coordination needs matrix. I then removed all cases where a role is not allowed to communicate with another role as defined in Equation 4.3. I calculated 16 alignment coordination matrices for APP, each representing coordination activities that took place for a certain set of requirements dependency and due to a certain reason like explained for the actual coordination matrices in Cataldo congruence.

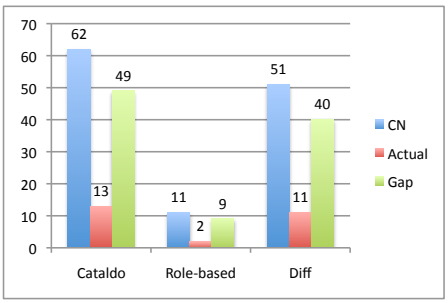
I calculated a *prime lack-of-coordination matrix* (Equation 4.5) to identify the real gaps by each combination of each set of dependencies and each reason of communication. I also identified false gaps (Equation 4.6) by subtracting each cell in the prime lack-of-coordination matrix from the role-coordination matrix (Equation 7.2.4). I finally calculated the backchannel communication (Equation 4.7). For each real gap, false gap, and backchannel communication instance, I gathered the same set of demographic information on the pair of members as for the gaps identified in Cataldo congruence.

Below I present the results of the application of the Cataldo and of the Role-based socio-technical congruence measures for the APP project. The results are, alternately, grouped by reason of communication, sets of dependencies, or roles performed by the team members. Differences between results obtained by applying Cataldo's and the Role-based measure are highlighted throughout the description of the results.

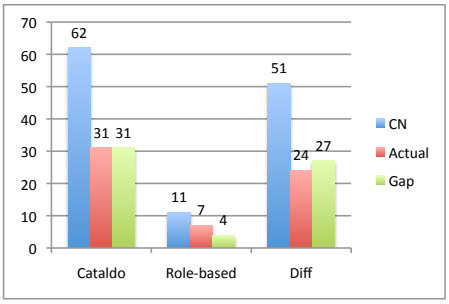
Coordination needs, actual communication, and gap numbers. I calculated the coordination needs, identified the actual (and aligned) communication, and computed the amount of congruence gaps for each set of dependent requirements network by reason of communication as previously explained. In Figure 7.7 I present consolidated results across the 4 sets of dependent requirements by reason of communication. For instance, in Cataldo I identified 18 coordination needs for the dependency set D_1, 12 for D_2, 16 for D_3, and 16 for D_4. Thus, Figure 7.7 shows the total of 62 coordination needs. The first cluster of columns from the left to the right side indicates the coordination needs (CN), actual communication (Actual), and gaps (Gap) calculated by Cataldo measure. The second cluster of columns indicates the coordination needs, aligned communication, and real gaps calculated by the Role-

based measure. The third cluster indicates the difference between Cataldo's and the Role-based's coordination needs, the amount of backchannel communication, and the amount of false gaps across the 4 sets of dependencies.

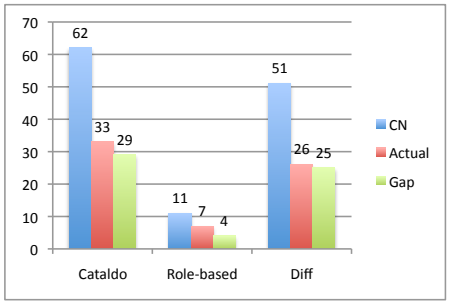
Overall, the high amount of backchannel communication indicates that the requirements dependencies created certain coordination needs that were attended despite the organization structure. Team members overlooked the formal structure in order to discuss what was necessary to accomplish work. On the other hand, the high amount of false gaps indicates that the lack of actual communication for certain coordination needs is between pairs of roles who were not expected to directly communicate with each other. This finding suggests that either the coordination needs may not have to be satisfied at all, or that team members used longer paths to communicate with each other. I.e., communication was intermediated by other colleagues. The investigation of longer paths was not conducted in this research, thus I cannot affirm which case applies here.



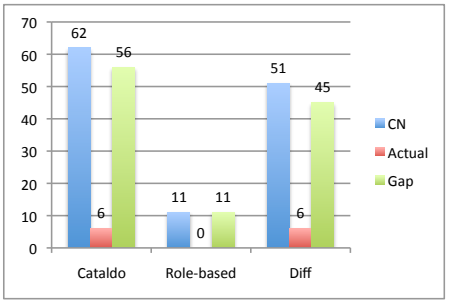
(a) Requirements Negotiation



(b) Requirements Clarification



(c) Communication of Changes



(d) Coordination of Activities

Figure 7.7: APP's coordination needs, actual coordination, and gap details

Chart 7.7(a) shows that there is a low number of actual (13) and aligned (2) communication compared to the number of expected coordination needs (62 in Cataldo's and 11 in the Role-based's) in the Requirements negotiation reason. Most of the ac-

tual communication (11 out of 13 instances) represent backchannel communication. In addition, most of the gaps identified by Cataldo are considered false gaps (40 out of 49). Only few instances are real gaps (9 out of 49). A similar pattern is observed in the Coordination of activities reason (see Chart 7.7(d)). There is a low number of actual (6) and no aligned (0) communication. The totality of actual instances of coordination (6 out of 6) are backchannel communication. The majority of the gaps identified by Cataldo are false gaps (45 out of 56) and only 11 instances are real gaps.

Chart 7.7(b) and Chart 7.7(c) display comparable patterns in the Requirements clarification and the Communication of changes reasons. Actual communication is higher in Cataldo than in the two above reasons. It represents about half of the expected coordination needs (31 out of 62, and 33 out of 62, respectively). Similarly to the previous reasons, a high amount of actual communication represents backchannel communication (24 out of 31 instances, and 26 out of 33 instances, respectively). Moreover, there is also a high amount of false gaps (27 out of 31 instances, and 25 out of 29 instances, respectively). Only 4 instances of gaps are considered real gaps for both reasons of communication.

Socio-technical congruence index. The level of alignment between the coordination needs and actual coordination was calculated for each set of dependent requirements and reason of communication network as presented above. Table 7.16(a) and Table 7.16(b) display the social-technical congruence indexes obtained by applying Cataldo and the Role-based measure, respectively. Congruence indexes by Cataldo range from 0 to 66%, and by the Role-based measure they range from 0 to 100%. On average, for both measures, the Requirements negotiation (RN) and Coordination of activities (CA) indexes are low, meaning that even when the organization structure is not considered in the Cataldo measure people satisfied very few of the coordination needs created by the requirements dependencies for both reasons. In contrast, the indexes are on average higher than 50% for the Requirements clarification (RC) and the Communication of changes (CC) reasons for both measures, indicating that more than half of the coordination needs were satisfied.

When contrasting Cataldo's to the Role-based congruence indexes, there is an increase for the set of dependencies D_1, D_2, and D_3 in the RC and CC networks and for the RN D_4 network in the Role-based measure. This increase indicates that actual communication in these networks are more aligned with the organizational structure than the actual communication in the remaining networks where there was a decrease

Table 7.16: APP' STC index by dependency and by reason of communication

(a) Cataldo

Dep	RN	RC	CC	CA
D_1	0.11	0.44	0.44	0.11
D_2	0.16	0.66	0.66	0.16
D_3	0.12	0.37	0.44	0.12
D_4	0.43	0.56	0.62	0
$Mean$	0.21	0.51	0.54	0.098

(b) Role-based

Dep	RN	RC	CC	CA
D_1	0	0. 67	0.67	0
D_2	0	1	1	0
D_3	0	0.50	0.50	0
D_4	0. 50	0.50	0.50	0
$Mean$	0. 12	0.67	0.67	0

on the index. One network is completely incongruent by Cataldo while there are 7 incongruent networks by the Role-based measure. Full incongruence (congruence index equal 0) means that no single coordination need is satisfied by actual communication. It does not mean that there are no actual communication. It is possible that communication took place between pair of roles that were not allowed to directly communicate with each other, what we call backchannel communication. Note that the 7 incongruence networks by the Role-based measure are the networks with the lowest congruence index by Cataldo. These are the RC (except for the dependent set D_4) and the CA networks.

Congruence gaps by roles. I further the investigation of requirements-driven coordination behavior by exploring demographic data about the identified gaps. I break down the gap information presented in Figure 7.7 (third column of each cluster) by roles. Each table entry indicates the pair of roles that the pair of members involved in the gap played in the project, and the number of gaps identified for the respective pair of roles across the 4 sets of dependencies. The *Gap* column indicates the gaps identified by Cataldo, the *RG* column indicates the Real gaps identified by the Role-based measure, and the *FG* column indicates the False gaps which were also identified by the Role-based measure. Note that in the *RG* column I indicate with an "x" the pair of roles that should not coordinate directly according to the imposed organizational structure. In other words, in the *RG* column I indicate an "x" in a table entry that maps the roles r_i and r_j when the position ij of the role-role matrix is equal 0 (refer to Chapter 4 for details). I show only the combination of pairs where there is at least one gap for any of the 3 types of gap.

In APP requirements are negotiated among business partners, project managers, and requirements analysts (ReqA). Development leaders (DevL) and the test leader

Table 7.17: APP's congruence gaps, real gaps, and false gaps

(a) Requirements negotiation

Roles	Gap	RG	FG
ReqA-Dev	2	x	2
ReqA-Tester	7	x	7
ReqA-TestL	0	0	0
Dev-ReqA	2	x	2
Dev-Dev	3	3	0
Dev-Tester	10	x	10
Dev-TestL	1	x	1
Tester-ReqA	7	x	7
Tester-Dev	9	x	9
Tester-Tester	5	5	0
TestL-ReqA	0	0	0
TestL-Dev	2	x	2
TestL-Tester	1	1	0
Total	49	9	40

(b) Requirements clarification

Roles	Gap	RG	FG
ReqA-Dev	2	x	2
ReqA-Tester	6	x	6
ReqA-TestL	0	0	0
Dev-ReqA	2	x	2
Dev-Dev	3	3	0
Dev-Tester	5	x	5
Dev-TestL	1	x	1
Tester-ReqA	6	x	6
Tester-Dev	3	x	3
Tester-Tester	0	0	0
TestL-ReqA	0	0	0
TestL-Dev	2	x	2
TestL-Tester	1	1	0
Total	31	4	27

(c) Communication of changes

Roles	Gap	RG	FG
ReqA-Dev	5	x	5
ReqA-Tester	0	x	0
ReqA-TestL	0	0	0
Dev-ReqA	5	x	5
Dev-Dev	3	3	0
Dev-Tester	10	x	10
Dev-TestL	1	x	1
Tester-ReqA	0	x	0
Tester-Dev	3	x	3
Tester-Tester	0	0	0
TestL-ReqA	0	0	0
TestL-Dev	1	x	1
TestL-Tester	1	1	0
Total	29	4	25

(d) Coordination of activities

Roles	Gap	RG	FG
ReqA-Dev	7	x	7
ReqA-Tester	4	x	4
ReqA-TestL	1	1	0
Dev-ReqA	7	x	7
Dev-Dev	3	3	0
Dev-Tester	11	x	11
Dev-TestL	1	x	1
Tester-ReqA	4	x	4
Tester-Dev	9	x	9
Tester-Tester	5	5	0
TestL-ReqA	1	1	0
TestL-Dev	2	x	2
TestL-Tester	1	1	0
Total	56	11	45

(TestL) are welcome to join the negotiation sessions when consensus about the project scope is not reached. Therefore, no coordination during requirements negotiation is expected among requirements analysis, developers, and testers. In addition, leaders (dev or test) are also not expected to conduct negotiations with developers or testers. The high number of gaps among these pairs of roles (49 out of 62 coordination needs)

reflects this process practice. Thus, it is not necessary to further analyze the gaps per type for this reason of communication (Table 7.17(a)).

Observe in Table 7.17(b) that the majority of the gap instances for the requirements clarification reason are false gaps (27 out of 31), meaning that the coordination needs that generated these gaps will never be satisfied because the individuals play roles that are not supposed to talk to each other according to the organizational structure. Although only 4 instances are real gaps, they indicate that over one-third of the coordination needs were not satisfied when the organizational structure was taken into account. These real gaps between developers (Dev-Dev equal 3), and test leader and tester (TestL-Dev equal 1) can be interpreted in two ways. One is that there was no need for developers or the test leader to request requirements clarification. A second is that developers and the test leader could not have their requests clarified because they could not communicate with other developers and the respective tester. Since the team was new to the product it is more likely that clarification requests were not communicated.

A similar pattern in present in Table 7.17(c). There are only 4 instances of real gaps but they represent over one-third of non-satisfied prime coordination needs according to the Role-based measure. These real gaps can either indicate that developers (Dev-Dev equal 3) and test leader and tester (TestL-Tester equal 1) did not need to communicate changes to each other, or that they did not receive communication about changes on requirements or related technical specifications. During the on site observations I had the chance to attend meetings where the requirements analysis and project managers were discussing and later approving requirements changes requests, thus I rather assume that the second is the most likely interpretation.

In the coordination of activities reason (see Table 7.17(d)), the totality of prime coordination needs are real gaps (11 out of 11). In other words, none of the coordination needs expected when considering the organizational structure were satisfied. Most of these real gaps are within members of the same team. For instance, developers did not coordinate activities with developers (Dev-Dev equal 3) and testers did not coordinate among themselves or with the test leader (Tester-Tester equal 5, TestL-Tester equal 1). This high lack of coordination is surprising since most of the members have cubicles assigned in the same office room than their teammates, and that the development and the test team separately performed daily short meetings aiming to synchronize information and to coordinate work-related activities.

Actual communication by roles. I complete the in-depth analysis of requirements-driven coordination patterns and their alignment with coordination needs and with the organizational structure by exploring demographic data about the identified actual communication. I break down the actual communication information presented in Figure 7.7 (second column of each cluster) by roles. Similarly to the gap analysis, each table entry indicates the pair of roles that the pair of members involved in the actual communication played in the project, and the number of actual communication identified for the respective pair of roles across the 4 sets of dependencies. The *Ac* column indicates the actual communication identified by Cataldo, the *Al* column indicates the Aligned communication identified by the Role-based measure, and the *BCc* column indicates the Backchannel communication which is also identified by the Role-based measure. Note that in the *Al* column I indicate with an "x" the pair of roles that should not directly coordinate according to the imposed organizational structure. In other words, in the *Al* column I indicate an "x" in a table entry that maps the roles r_i and r_j when the position ij of the role-role matrix is equal 0 (refer to Chapter 4 for details).

Table 7.18 reveals that the majority of the actual communication across the 4 reasons (67 out of 83, over four-fifths) is considered backchannel communication according to the Role-based measure. In other words, the majority of the reported actual communication took place between roles that were not expected to directly communicate with each other. There is a balance between three roles as initiators of these occurrences of backchannel communication, as follows: requirements analysts (ReqA) initiated 23 instances of backchannel communication, tester initiated 23 of them, and developers initiated 20 instances. The 1 remaining instance was initiated by the test leader (TestL). However, when I group these roles by team one can see that the test team is predominant (24 out of 67), followed by the requirements analyst team (23 out of 67), and last by the development team (20 out of 67). In contrast to the high amount of backchannel communication, Table 7.18 shows that there are few instances of aligned communication (16 out of 83). Testers were the ones who most initiated aligned communication (10 out of 16), followed by requirements analysts and the test leader (3 instances each role).

More specifically, Table 7.18(a) shows that over nine-tenth (10 out of 11) of the backchannel communication about Requirements negotiation took place between requirements analysts and developers (ReqA-Dev equal 5, Dev-ReqA equal 5), and that the only 2 instances of aligned communication were between a requirements analyst

Table 7.18: APP's actual, aligned, and backchannel communication

(a) Requirements negotiation

Roles	Ac	Al	BCc
ReqA-Dev	5	x	5
ReqA-Tester	0	x	0
ReqA-TestL	1	1	0
Dev-ReqA	5	x	5
Dev-Dev	0	0	0
Dev-Tester	1	x	1
Dev-TestL	0	x	0
Tester-ReqA	0	x	0
Tester-Dev	0	x	0
Tester-Tester	0	0	0
TestL-ReqA	1	1	0
TestL-Dev	0	x	0
TestL-Tester	0	0	0
Total	13	2	11

(b) Requirements clarification

Roles	Ac	Al	BCc
ReqA-Dev	5	x	5
ReqA-Tester	1	x	1
ReqA-TestL	1	1	0
Dev-ReqA	5	x	5
Dev-Dev	0	0	0
Dev-Tester	6	x	6
Dev-TestL	0	x	0
Tester-ReqA	1	x	1
Tester-Dev	6	x	6
Tester-Tester	5	5	0
TestL-ReqA	1	1	0
TestL-Dev	0	x	0
TestL-Tester	0	0	0
Total	31	7	24

(c) Communication of changes

Roles	Ac	Al	BCc
ReqA-Dev	2	x	2
ReqA-Tester	7	x	7
ReqA-TestL	1	1	0
Dev-ReqA	2	x	2
Dev-Dev	0	0	0
Dev-Tester	1	x	1
Dev-TestL	0	x	0
Tester-ReqA	7	x	7
Tester-Dev	6	x	6
Tester-Tester	5	5	0
TestL-ReqA	1	1	0
TestL-Dev	1	x	1
TestL-Tester	0	0	0
Total	33	7	26

(d) Coordination of activities

Roles	Ac	Al	BCc
ReqA-Dev	0	x	0
ReqA-Tester	3	x	3
ReqA-TestL	0	0	0
Dev-ReqA	0	x	0
Dev-Dev	0	0	0
Dev-Tester	0	x	0
Dev-TestL	0	x	0
Tester-ReqA	3	x	3
Tester-Dev	0	x	0
Tester-Tester	0	0	0
TestL-ReqA	0	0	0
TestL-Dev	0	x	0
TestL-Tester	0	0	0
Total	6	0	6

and the test leader (ReqA-TestL equal 1, TestL-ReqA equal 1).

A closer look in Table 7.18(b) reveals that the majority of backchannel communication (22 out of 24) about Requirements clarification involved a developer, being this role either the initiator or the target of the conversation. It also shows that the test team went to the requirements analysts to clarify requirements. Two instances of these clarifications (Tester-ReqA equal 1, ReqA-Tester equal 1) are considered backchannel communication, and two of them satisfied coordination needs (TestL-ReqA equal 1, ReqA-TestL equal 1). Testers initiated 5 (out of 7) instances of aligned communication.

In Table 7.18(c) see that over two-thirds (18 out of 26) of the backchannel communication about Communication of changes involves a requirements analyst (ReqA-Dev equal 2, ReqA-Tester equal 7, and their symmetric pairs). The remaining instances of backchannel communication took place between the test and the development teams (TestL-Dev equal 1, Tester-Dev equal 6, Dev-Tester equal 1). Like in the Requirements clarification reason, testers initiated 5 (out of 7) instances of aligned communication.

Table 7.18(d) shows that none of the actual communication instances is aligned with the organization structure in the Coordination of activities reason. Hence, the 6 instances of backchannel communication (Tester-ReqA equal 3, ReqA-Tester equal 3). No other communication related to coordinating activities was reported.

7.2.5 Summary of Empirical Insights

In order to highlight the main insights gained in the APP case study, I summarize the insights by research question and measure in Table 7.19. Additional measures to the initial set of measures proposed in the framework are in bold.

Table 7.19: APP' summary of measures by RDC aspect

RCSN Characterization		
Measure	*Question*	*Empirically-informed insight*
RCSN visualization	How does the RCSN look like?	Emergent and isolated members
RCSN size	How many people collaborating?	About 1/3 are emergent
RCSN density	How tightly-coupled is the RCT?	1/5 of communication took place
Ties statistics	What kind of interactions?	High cross-sites and cross-teams
RCSN Structure		
Measure	*Question*	*Empirically-informed insight*
Centralization	Is collaboration centralized around certain members?	Decentralized collaboration
Core-periphery	Who is at the core of the RCSN?	Requirements analysts and testers
Ties reciprocity	Do members collaborate in a equal manner with each other?	Low reciprocal communication, high awareness reciprocity
Clique	Who are the well-connected groups within the RCSN, if any?	One representative of each team
RCSN Information flow		
Measure	*Question*	*Empirically-informed insight*
Component	Are all members of the RCT connected?	There are disconnected members
Reachability	To what extent information can be shared with everyone?	Emergent cannot reach others
Cutpoints	Are there members that if removed would disrupt info flow? Are there members that if removed would disrupt info flow?	Mainly assigned members except requirements analysts
Degree	Who are the central members?	Mostly testers
RCSNs Alignment		
Measure	*Question*	*Empirically-informed insight*
Alignment	To what extent are social relationships aligned to reqs dep? To what extent RCTs follow the organization structure?	Little alignment Mostly disaligned

7.3 Threats to Validity

Threats to validity are influences that may limit one's ability to interpret or draw conclusions from the study's data [106]. Ignoring the threats can lead to the wrong conclusions regarding the validity of the results. For example, a practitioner might assume that the results would apply to his situation where the external validity could indicate problems regarding generalizability [75]. Despite the fact that there are threats to the validity of my multiple empirical case study results, I found initial but nonetheless useful empirical observations regarding requirements-driven collaboration. In this section I present the threats to validity for the two investigated projects and discuss the actions I took to mitigate the potential weaknesses of the empirical study design.

Construct validity. I adopted a restricted conceptualization of coordination to construct the social networks. I conceptualized it by the one-to-one interactions established by team members when communicating about the project. Communication is the predominant coordination mechanism investigated in software engineering, but one knows from literature that team members also coordinate by following pre-defined processes [132] which are usually supported by tools. For instance, team members coordinate work by using comments from source code repositories (e.g., [24], [42]) and issues tracking tools (e.g., [43], [52], [136]). During the three months I spent on site I observed that APP and SHIP mostly coordinated using communication media such as e-mail, instant message, and telephone, or by communicating face-to-face. Coordination through communication took place in individual and group levels, when one team member approached each other and group meetings occurred, respectively. I believe that the coordination through pre-defined processes was not predominantly noted for the APP project because the majority of the team members were collocated in the same office building, thus face-to-face communication was facilitated. For the SHIP project, I believe that the team members' familiarity with the product and source code excused them of using source code versioning or bug tracking tools to coordinate. Based on the observations I believe that the restricted conceptualization of coordination is not detrimental to the study.

In my study I assumed that a single message from a sender (or source) member S was communicated to a receiver (or target) member R. However, this may not be the case as in the data collection I only collected information about communication ties

between pairs of people, but not about which exact message they were communicating about. Therefore, a communication path may allow the sending of information by S but does not necessarily imply that the same information was received by R. However, I did collect information about the specific reason why communication took place (e.g., communication of changes, requirements clarification) and this indicates that channels that I constructed over those pairs of members were most likely to convey the desired information type.

Team distribution is a concern in the application of social network analysis measures that use the concept of group to calculate the measure outcome. In my study the *brokerage* measure grouped members per requirement they were assigned to work on (dependee, dependent, both requirements groups). I was unaware of the requirements' group size limitation at the time I selected the case studies, otherwise I would have aimed for projects that better satisfied this condition.

The conceptualization of awareness in the case studies was limited. Questionnaire questions for three types of awareness were asked regarding the project level instead of being formulated to collect requirements-driven data, which are: general awareness, availability, and familiarity. For example, I asked about familiarity *"How closely did you work with this person on your last project together?"* instead of asking *"How closely did you work on this certain requirement with this person on your last project together?"* because I was not interested in past requirements. In fact, there is no past history about actual requirements. Thus, I consider that these awareness types are general to the project, and cannot be customized to provide requirements-oriented information. However, I did constructed project-centric social networks for the three awareness types, and used this overall knowledge to better understand how team members exchange information and share knowledge relevant to coordinate requirements-driven work.

Reliability. Specifically in terms of the examination of the alignment between networks (research question 5), the study of socio-technical congruence depends on the conceptualization of the social and technical dimensions, on the context of the project being investigated, and on the data sources available for analysis. I believe that variations from project to project make difficult the replication and comparison of results among different studies. Even though, I share rich details of the empirical case study design aiming to provide insights and inspire others who intend to conduct socio-technical investigation in similar settings. Despite the dependability, I understand

that case studies like mine help to further the understanding of coordination behavior in software teams and contribute to the development of a body of knowledge about requirements-driven collaboration.

Internal validity. Social networks are dynamic. Longitudinal studies with multiple data collection points to construct the social networks over time may provide the researcher with valuable indications of changes on coordination patterns. During the observation sessions I noticed well-established practices, such as same communication media were used to contact certain members, meetings and discussions kept the same periodicity throughout the three months, and team members usually consulted same people for help or clarifications. These practices are an indicator that the behavior is likely to be constant or that variations may not present discrepant results for both projects, and the absence of a longitudinal study is potentially not prejudicial. Therefore, I believe that the single data collection point to construct the social networks in this study is acceptable. I believe that the rich contextual data I provide here contributes to minimize the impact of one single data collection point.

The self-reported communication data collected through the questionnaire is another internal validity threat. Despite the awareness that questionnaires are occasionally viewed as unreliable data sources to collect social network data since they are based on the memory and perceptions of the participants [29], the lack of access to any software repository or automatic logs generated by the tools used by the APP and SHIP teams, and the need to collect a large amount of data led me to decide that a questionnaire was the most reliable data collection method to be adopted. The questionnaire was deployed in the beginning of the Testing phase, when all team members were still actively working in the project. The time frame of the questionnaire application may have decreased the chances that team members forgot to report instances of communication but it is dangerous to assume that they did not forget instances. To deal with this threat, I used a limited secondary data collection method to verify the accuracy of the communication questionnaire data. I collected communication data through a work diary, which is an instrument used to record various events that occur during the respondent's day. I asked each team member of both projects to daily record details of his requirements-related communication in a provided personal diary but unfortunately only a small percentage of the research participants filled out the diary for a significant and sequential set of days. For these members, I compared the frequency of communication and the members that one

communicated with in the both data collection instruments and identified an almost perfect match between responses on both data sets. Consequently, I trust that the self-reported communication questionnaire data is reliable.

On the other hand, questionnaire awareness data used to construct the social networks may have suffered from misunderstandings. Different team members may have interpreted awareness levels in different ways based on their personal perception. In addition, team members may have felt apprehensive of reporting lack of awareness when this could be seen as a sign of work underperformance, especially in offshoring relationships where maintaining business partnerships is very important. To check the accuracy of self-reported awareness data, during interviews I asked team members questions about their overall levels of awareness of what others were doing and about how they become aware of their colleagues. Although this was not a point-to-point verification, interview results indicated that self-reported data were aligned with the members' awareness perceptions.

Social network analysis is especially sensitive to missing data since models of network structures assume that complete information is available on the relations to be described [20]. A missing answer does not necessarily imply the absence of a social tie. For instance, people may refuse to answer a particular question. In this study, there are two kinds of missing data: from non-respondents and from non-participants. Out of the 21 research participants, only 2 did not reply to the questionnaire, all of them from the APP project. To deal with missing social network data from non-respondents, I used the *available-case analysis* technique [125] where the missing tie from member B to member A (tie $B - A$) is fulfilled with the value indicated by member A (tie $A - B$). Out of the 59 invited team members, 4 rejected participation, all of them from the US members from the SHIP project. The totality of the members who rejected participation were cited by respondents as people they communicated with or were aware of, thus they were included in the social networks. The non-participants were identified as emergent members to the networks because although they were allocated to work in the projects they were not assigned to work on the set of dependent requirements covered in my study. Thus, since non-participants are emergent members I did not replace their missing data. Their participation on the network was not expected in the first place, thus I believe that using the available-case analysis technique (or any other technique) would mislead the networks collaboration behavior.

External validity. I investigated two projects only. This limitation constrains the generalizability of the empirical case studies insights but it does not stop me of contributing to scientific development by accumulating knowledge about the requirements-driven collaboration topic and in the socio-technical congruence field of study. To minimize the impact of this limitation, I selected projects with distinct characteristics, such as group size (14 members in SHIP and 45 members in APP) and physical dispersion among team members (SHIP is distributed between the US and Brazil, and APP is collocated in a single country (Brazil), except business partners are located in the US, but the Brazilian team is distributed in three different buildings about 5-10 minutes away from each other). Both projects are responsible for the maintenance of legacy products. This characteristic may limit the results to maintenance projects only. To eliminate this dependency to the type of project, I plan to conduct a similar case study in a near future in new software development projects. Refer to the next section for details.

Another limitation towards generalizability is the small number of requirements in each dependency set investigated in each project. The mean of requirements in each set is 2.5 in both projects. A large number of requirements by set may change collaboration behavior of the requirements-centric teams. The number of dependent requirement sets was also small for both projects. Even though 18 requirements were identified in SHIP and 20 in APP, only 6 and 5 requirements, respectively, were involved in the set of dependencies identified. This small number have limited the examination of requirements-driven collaboration in each project as a whole to interactions that took place around these dependent requirements. I believe that the fact that most of the project members are involved somehow with the requirements-centric social networks (either as an assigned or as an emergent member), individual team member behaviors were captured minimizing the impact of the requirements representativeness. I could not anticipate the number of requirements by dependency set or the number of requirements dependencies prior to the data collection phase, otherwise I would have aimed for projects with a higher number of requirements per requirement sets and of dependency sets itself. However, I believe that these small samples do not compromise the validity of the empirical insights since the main goal of the empirical study was to illustrate the applicability of the framework itself.

Chapter 8

Revisiting the Research Questions

The goal of this research was two-fold. First, to develop an approach to examine requirements-driven collaboration in which a set of techniques to perform the examination of communication and fleeting knowledge underlying collaboration driven by requirements is part of the approach. Second, to further the understanding of requirements-driven collaboration by empirically examining communication and fleeting knowledge in the coordination necessary to the development of requirements in industrial projects. To meet this research goal I have answered the formulated research questions. In this chapter I discuss my answers to these questions.

8.1 A Framework to the Study of Requirements-Driven Collaboration

Research question 1 asked *"How can one investigate requirements-driven collaboration?"*. This question has been answered by the development of the framework for studying requirements-driven collaboration. The development of the framework was seeded in literature review, and developed incrementally through its application to the design of the field studies. More specifically, it was inspired in the work of Cataldo et al.[24], and Erhlich and Chang [48] which uses social network theory to investigate collaboration behavior in software development teams. The incremental development process that followed was based in empirically-informed insights obtained through the framework application in a multiple case study of requirements-driven collaboration.

The framework proposes an approach that uses concepts and measures from social network analysis to obtain insights about the communication and fleeting knowl-

edge patterns of those involved in requirements-driven collaboration. The approach is based on the concept of a requirements-centric team, which is a cross-functional group whose members' work activities are related to one or more interrelated requirements, as well as downstream artifacts such as design, code and tests. The requirements-centric team concept also applies to non-functional requirements, although their definition may be more lose and abstract in terms of who is involved with them. In the investigated cases there was no non-functional requirements, thus this is one assumption that has to be examined for applicability.

To analyze the collaboration within RCTs, a requirements-centric social network was defined. A requirements-centric social network is a social network [135] that represents the members (also called actors) and relationships (also called ties) in a requirements-centric team. The actors in a requirements-centric social network are among the members of the requirements-centric team, and the ties in the network are representations of different relationships during these members collaboration.

The framework also consists of a number of measures from social network analysis defined as mechanisms to explore requirements-driven collaboration. An initial set of measures was defined based in literature review. More social network analysis measures were incrementally selected and added to the initial set of measures as the empirical investigation of requirements-driven collaboration in the multiple exploratory case study progressed. The selection of these additional measures was based on a thorough decision process. More finer-grain questions within each posed research questions were raised as a result of the incremental data analysis and the findings review process with the team members. The need to identify which measure would answer each of the raised finer-grain questions led me to further knowledge about social network analysis and to select additional measures. The final list of measures that compose the framework is presented in Table 8.1. The complete version of the framework is found in Appendix D.

In its current structure, the framework provides sufficient flexibility to be extended in terms of additional social network analysis measures and of requirements-driven collaboration aspects to be investigated. The framework adoption by researchers and practitioners has to be followed by the interpretation of the results based on contextual information or demographic data about the requirements and team members. This kind of interpretation is inherent and required in the analysis of social networks [117].

Table 8.1: Final list of measures that compose the framework to study requirements-driven collaboration

Characterization of Communication and Fleeting Knowledge in RDC		
Measure	*Question*	*Empirically-informed insight*
RCSN visualization	How does the RCSN look like?	Overall RDC insights
RCSN size	How many people are collaborating?	Number of members
RCSN density	How tightly-coupled is the RCT?	Cohesion
Ties statistics	What kind of interactions are present?	Patterns of collaboration
Communication and Fleeting Knowledge Network Structures in RDC		
Measure	*Question*	*Empirically-informed insight*
Centralization	Is collaboration centralized around certain members?	Knowledge distribution
Core-periphery	Who is at the core of the RCSN?	Key members
Ties reciprocity	Do members collaborate in a equal manner with each other?	Reciprocal collaboration
Clique	Who are the well-connected groups within the RCSN?	Key members
Information Flow Patterns in RDC		
Measure	*Question*	*Empirically-informed insight*
Component	Are all members of the RCT connected?	Extent of network division
Reachability	To what extent information can be shared with everyone?	Ability to reach all members
Cutpoints	Are there members that if removed would disrupt info flow?	Crucial members
Degree	Who are the central members?	Most active members
Brokerage	Who are the mediators of information?	Communication brokers
Alignment among Coord. Needs, Actual Coord., and Org. Structure in RDC		
Measure	*Question*	*Empirically-informed insight*
Alignment	To what extent are social relationships aligned reqs dep? To what extent RCTs follow the organization structure?	Ability to coordinate aligned with requirements dependencies Coordination behavior aligned organization structure

8.2 Characteristics of Communication and Fleeting Knowledge in RDC

Research question 2 asked *"What are typical characteristics of communication and fleeting knowledge networks established during collaboration driven by requirements?"*. This question has been answered by empirically examining the communication and the fleeting knowledge RCSNs in requirements-driven collaboration. To examine the communication and fleeting knowledge RCSNs, I have applied the framework to two industrial projects. Requirements-centric social networks were constructed for each set of requirements dependency. By analyzing each RCSNs, I have identified a number of patterns discussed below.

Communication-RCSNs vary in membership and amount of communication exchanged between their members. These networks also have isolated members who did not collaborate with others. In order to develop a broad understanding of requirements-driven collaboration patterns in each project, I visually displayed each of the communication requirements-centric social network. This visual representation is called a sociogram. By inspecting the sociograms I found that the amount of communication and people involved in each network varied per topic discussed. People who were not assigned to work on the requirements were consulted by assigned members, indicating a dynamic aspect of collaboration driven by requirements. Interestingly, members who were assigned to work on the requirements were found isolated from any other requirements-centric team member. Isolated members were more frequent in the APP project, suggesting that these members did not collaborate in a one-to-one manner with others working on the same requirements. The fact that these members were new hires to the company and new to the project brings implications for collaboration among new teams. As people develop a relationship and trust each other, it is common for them to contact colleagues for asking for help or clarifications. It is common to familiar members to exchange information during informal meetings, such as a talk down the corridor or in the cafeteria. When new members do not know who has expertise on each area, or who is doing what in the project, they may not know who to go to for help or clarifications. Then, these people may feel disconnected to the team or even be left out discussions for 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 the team.

Communication-RCSNs are dynamic, and included important cross-site and cross-team interactions. By calculating the number of members in each network and identifying statistics about the ties I found that there were more people collaborating during the development of each dependent requirements than was originally planned, and on average over one third of interactions in the team across the reasons of communication were with emergent team members. Whereas a previous study of one distributed team by my research lab colleagues [39] has identified that teams working on same requirements are dynamic, here I bring evidence that suggests that teams working on dependent requirements follow the same pattern. These members emerged mainly because they are collaborators of the project who are not assigned to work in development tasks, such as the logistic manager who belongs to the SHIP IT team associated with one of the ORG's manufacturing plant unit. Among the emergent members there is also people who have worked in the past in the project. These former team members still support the current team when necessary. For example, a former US development leader provides technical support about the project architecture or about component interfaces with other systems when invited by the current development leaders. Interestingly, although there are emergent members, isolated people were identified through the inspection of the RCSNs sociogram. These isolated people were in fact assigned to work on the requirements but for some reason they did not report communicating in a one-to-one manner with colleagues. However, in my visit on site I did attend meetings where these isolated members were sitting together with the remaining of the team meaning that they collaborated somehow to the discussions and the development of the requirements.

Requirements clarification (RC) and communication of changes (CC) were the most-often discussed topics among requirements-centric team members. For each project, the most-often-reported interaction between assigned members by communication-RCSN was about requirements clarification. In the SHIP project, most of the requirements clarification interactions were initiated by members of the test team because of their concern of clearly comprehending the project requirements in order to ensure that the team was correctly validating and verifying the product against the requirements specifications. In the APP project, mostly testers requested to the requirements analysts clarifications about the requirements.

These fine-grained results are empirical evidence that testers need support from the remaining of the team to understand the project requirements and more efficiently perform their tasks. Approaches such as test-driven development [11], where a requirement has to be clearly understood before the test case is written, can be adopted to impose the comprehension of the requirement 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. Preliminary 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 [51].

In addition, the most frequent interaction with emergent members was about communication of changes. Although changes consisted of such a large proportion of both intra-site and cross-site communications, 62.5% of the respondents believed that they did not receive requirements in a timely fashion. This highlights an area in which improved change awareness, by increasing its timeliness, may have a strong effect on collaboration. Future research should pursue the development of improved methods for communication of changes, both to improve the timeliness of change notifications and to reduce the communication overhead required to make team members aware of every change.

Awareness-RCSNs are also dynamic in terms of membership, and included important cross-site and cross-team interactions. Team members were aware of members who were not assigned to work on the requirements. Interestingly, for the SHIP project, most of the emergent people that members were aware of are located in the US office. This suggests that the ability of the SHIP's team members of staying aware of colleagues was not affected by distance. SHIP is a mature team in terms of the expertise on the product and in working together. The team has been developing working practices together for over three years, and each and every member in Brazil have had the change to travel to meet the US teammates. This accumulated experience is likely to have helped the team to maintain awareness even of those in remote locations. Surprisingly, APP's team was also aware of emergent members characterizing dynamic awareness-RCSNs. This team was new to the applications and members were not familiar with each other. Physical proximity may have helped the members to maintain awareness of colleagues. For both projects a large amount of cross-sites and cross-teams awareness was reported, indicating that team members

have knowledge of what colleagues working on different locations and teams are doing that is related to one's work. The teams maintained awareness beyond geographical and organizational boundaries.

In addition, it was also observed that on average awareness networks are denser than communication networks, meaning that at that given time in the project team members of both projects were using other means to maintain awareness than communication. About one-third of team members of each project reported in the questionnaire that they maintain awareness of what others are doing that is related to their work through task assignments or project documentation. Changes may affect the ability of maintaining up-to-date awareness through these means since at ORG tasks are assigned once a week and documentation has to be formally approved before changes are announced to the team.

Requirements-centric teams did not communicate in their full capacity or did not stay aware of each other colleague working in the dependent requirements. When I examined the amount of communication within each RCSN (density) I found that in SHIP, on average, less than one-fourth of the communication load that the requirements-centric teams could handle in fact took place. Densities increased when I removed the emergent members suggesting that team members assigned to work on the requirements communicated more with each other than when considering the emergent members. It is difficult to conclude whether this amount of communication was optimal or deficient, but since the project succeeded by deploying the requirements on time and on budget, I would assume that it is safe to claim that this amount of communication was sufficient. In APP this pattern is repeated. About one-fourth of the communication that could have taken place in fact took place. However, in APP the weekly and daily meeting mechanisms may have diminished the team's need to communicate in a one-to-one manner.

The same pattern was found when I examined the density of the awareness-RCSN networks. Team members are aware of most of their colleagues when emergent members are ignored. This finding with respect to awareness in RCSNs also have important implications for requirements-centric collaboration and design of collaborative systems. A team member generally knew who else was working on the same dependent requirements, regardless of geographical location. This was somehow surprising, given the RCSNs dynamic nature, and the significant number of emergent relationships in these networks. In SHIP, the potential lack of impact of distance on the collaboration

within these dynamic RCSNs may have been eliminated or minimized by the team expertise on the product and experience working together as discussed above. In APP, not only the physical proximity but also the frequent periodic group meetings may have collaborated for the team members to maintain awareness of colleagues.

Table 8.2: Summary of findings about RCSNs characterization by project

Finding	**SHIP**	**APP**
Emergent members are present	Many	Many
Isolated members are present	Very few	Many
Variation on amount of communication	Low	Mid
Dynamic RCSNs	Yes	Yes
RC and CC are the most discussed reasons	Yes	Yes
Cross-sites communication is predominant	No	Yes
Cross-teams communication is predominant	No	Yes
Cross-sites awareness is predominant	No	Yes
Cross-teams awareness is predominant	No	Yes

Table 8.2 summarizes the main insights gained about the characterization of the RCSNs per project.

8.3 Communication and Fleeting Knowledge Network Structures in RDC

Research question 3 asked *"What structures do communication and fleeting knowledge networks have in requirements-driven collaboration?"*. This question has also been answered by empirically examining the structure of each of the communication- and the awareness-RCSNs in the two industrial projects.

Knowledge distribution is not concentrated in the hands of few members, as well as current awareness is not a privilege of few members. By calculating the RCSN centralization index I found that SHIP has more decentralized network structures where more team members are involved in exchanging information with each other. The power of knowledge distribution is not in the hands of few members, which diminishes the risks of having communication breakdowns because of information overload. This finding supports previous understanding that requirements-centric teams composed of cross-functional roles bring different knowledge about the requirement, and thus contribute to the project development as a

whole. The awareness-RCSNs follow the same pattern. The ability to stay aware of others is not concentrated in the hands of few members suggesting that this team is well informed of what is going on in the project that may affect each other. When the distribution of updates and changes is not concentrated in few people the tendency is that the team will be less affected or disturbed by the sudden absence of a colleague.

The network structures for the APP project follow the same pattern. They are more decentralized for the communication networks and very decentralized for the awareness networks. This finding contradicts the organization design imposed by APP's senior management. They created a structure intended to centralize distribution of information and control awareness of decisions made on the hands of the project leaders, being them the requirements analyst leader, the 5 development leaders, and the test leader. Later, the Role-based socio-technical congruence measure showed the extend of misalignment between actual coordination behavior and the organization structure.

Communication and awareness are centered in key members according to the team expertise characteristics. I found that a structure of a core and a surrounding periphery is not predominant in the SHIP communication requirements-centric social networks, meaning that information exchange is not focused in few members. However, for the coordination of activities networks with a strong indication of a core-periphery structure I found that the core is composed of the development leaders and two senior developers, one from Brazil and one located in the US. These members have not only expertise on the product but they were also closely controlling the tasks allocation and dependencies with external partners aiming to avoid reword or breakdowns. Although the awareness networks were found with a more strong indication of a core-periphery structure, the peripheral members are in totality emergent members. These emergent people were not expected to be aware of their colleagues since they were not assigned to work on the requirements. Thus, it is likely that the awareness networks do not have a core-periphery structure, supporting the centralization results. Once again, this finding indicates that SHIP is a team that has a more flat structure supporting the members collaboration.

In contrast to the SHIP project, in APP a strong core-periphery structure is predominant for both the communication and the awareness networks. The core of the networks was formed mainly by requirements analysts and testers, which is explained by the need members had to get familiar with the project.

The low amount of reciprocal collaboration driven by requirements did not stopped the teams to delivering the project on time and as expected by the customers. The low reciprocity indexes suggest that information is not mutually exchanged in the requirements-centric social networks. However, a more finer-grain analysis of which pair of members are involved in reciprocal communication based on visual inspection of the sociograms revealed that assigned members are to a certain extent engaged in reciprocal relationships. This finding brings implications for knowledge sharing and information exchange processes. Previous research has found that team members are more likely to share knowledge when engaged in reciprocal relationships [45]. Thus, it is imperial that organization structures and managers support the establishment of reciprocal relationships between team members by facilitating accessibility and availability of colleagues, specially remote members.

More specifically, in SHIP I found that there are more pairs of reciprocal communication interactions across the networks than pairs of team members who got involved in reciprocal relationships. This indicates that despite the low indexes of reciprocity, when communication takes place it is likely it will be returned back by who is receiving it. More specifically, I found that communication of changes is the reason that most have reciprocal ties, suggesting that when a team member notified a colleague of a change the colleague communicated back. This equality in exchanging information indicates that team members do collaborate in propagating notifications of changes, and implies that the team mobilize itself together to take actions to address these changes.

In the APP project, although there is no patterns across all networks, the Requirements negotiation and the Requirements clarification networks have higher indexes indicating that more people got involved in reciprocal communication and more of the actual communications are reciprocal. On the other hand, the Communication of changes and Coordination of activities networks have lower indexes, indicating that changes were mostly communicated and no reply was provided back, and that activities were coordinated in a more directed way, for instance, from the leaders to the developers and testers, than discussed between the involved parts. Since the senior management intentionally designed the team in a way that leaders would be in charge of leading the development activities, this result supports that the team acted as expected for these two types of communication. Despite both project have low reciprocity indexes, they delivered the new version of the products on time and attended the scope agreed with the customers. This suggests that reciprocal collaboration is

not premise for successful requirements-driven collaboration.

Fully connected informal subgroups within the networks are formed according to the team expertise characteristics. In SHIP, the many weak cliques found revealed that people group together corresponding to the topic of discussion. For example, in the requirements negotiation reason cliques are formed by the development leaders, the test leader, and the project manager. They more intensively discuss which requirements should be part of the project scope than other members. This pattern is associated with the pre-defined roles and responsibilities established in the project, and indicates that the team closely behaves in alignment with the organization structure.

The APP networks are characterized by few cliques. When cliques were found, they were composed by a requirements analyst (ReqAn), a developer, and a tester for the Requirements clarification networks. This pattern reflects what I had the change to observe 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 requirement. Since developers spent three months working on reverse engineering to document the applications that APP was developing in the investigated project release, it was natural for the unfamiliar testers to seek for the developers' help to learn about the requirements. The Communication of changes cliques are formed by a developer and two testers. This finding would be somehow intriguing if I have no knowledge of the project context. One would expect that requirements analysts are involved on these cliques because they were very active in negotiating requirements changes and communicating them to the project managers and developers. However, the enforced organization structure assigned to the developer leaders the responsibility of notifying changes to their teammates. It is likely that these notifications took place in group meetings and then the developers forward this information in person to the testers. In fact I had the change to observe three episodes where this was the flow of information exchange about changes to requirements.

Table 8.3 summarizes the main insights gained about the structures of the RCSNs per project.

Table 8.3: Summary of findings about RCSNs structures by project

Finding	**SHIP**	**APP**
Communication structure	Decentralized	Decentralized
Awareness structure	Decentralized	Decentralized
Core-periphery structure	Absent	Present
Core members are ...	US Dev leader	ReqAn, Testers
Reciprocal ties are present	Not predominant	Not predominant
Cliques are present	Many	Few
Cliques are formed among ...	Leaders and devs	Team representatives

8.4 Information Flow Patterns in RDC

Research question 4 asked *"How do team members exchange information and share knowledge relevant to performing work driven by requirements? In other words, how does information flow within a requirements-centric social network?"*. This question has been answered by empirically examining the communication-RCSNs.

Information cannot be propagated to all team members. More specifically, developers are the predominant members that others cannot reach. In the SHIP project, most of the communication-RCSNs are connected, all the majority of the team members could communicate with each other. Although the strong component test yielded multiple components when the direction of the ties were considered, most of the members that cannot reach others are the emergent members. This finding is not surprising since most of the emergent members did not have the chance to report back about their communication interactions with colleagues. In contrast, the communication-RCSNs for the APP project are mainly disconnected. There are isolated members who did not communicate with others. The majority of these members are developers who were assigned to work on the requirements. I observed these members collaborating with colleagues during my visit on site. Thus, it is possible that these members only did not participate in one-to-one communication interactions. Despite the reason, this finding brings to the attention of managers that newcomers may not get engaged in individual conversation as expected for developing their tasks.

Development team members were the people who most shared knowledge, and leaders were the members most sought by others. The degree centrality

measure identified that development team members where the members who most shared knowledge, and leaders were the members most sought by others in the SHIP project. More specifically, the US development leader and an US senior developer were the ones who most communicated with others. These members are the two people who are longer working for SHIP, and have direct access to the business partners. Interestingly, they were not assigned to work on both requirements, suggesting that expertise on the product and familiarity with others was what made them so active in the project and not the fact that they worked in implementing both requirements in the dependency sets. In the APP project, testers were the most predominant members sharing information with others. This result is somehow surprising since the developers were the ones who had spent more time working in the applications to learn about them through the reverse engineering activity, and were expected to be the experts on the project.

Development leaders and testers may disrupt exchange of information necessary for requirements-driven collaboration if removed from the teams. In the SHIP project, the cutpoint measure revealed that *AC*, the US development leader, is a cutpoint for each of the communication networks. This indicates that AC is a crucial member in the project regarding any topic and focus of discussion. She plays not only the development leader, but she is also the most knowledgeable members currently assigned to work on the project. She supports the project leader and acts as the focal point with business partners. The project depends on her. On the other hand, in APP testers are the predominant cutpoints for the communication networks. This is a surprising finding since testers were mostly involved with testing activities and rarely got involved in supporting other roles in defining requirements or managing the project. However, most of the communication-RCSNs have a cut-set, indicating that it is necessary to remove more than one member to disrupt the exchange of information among the requirements-centric team members.

By using the customized brokerage measure that I proposed, I found the presence of brokers in the SHIP's communication-RCSNs. Their presence in any of these networks indicates the existence of pairs across two dependent requirement networks that do not communicate directly and for which the information flow was mediated. In the context of requirements engineering, this indicates that those members whose communication was brokered did not use and possible had no direct way of communicating requirements information. Misunderstandings or information loss could occur

in these mediated interactions. I have also found interesting patterns of members brokering information between a pair of colleagues. I discuss these patterns here.

Brokerage is predominant in certain reasons of communication. Brokers of information flow were identified in eighteen out of the twenty-five RDSNs I investigated. A pattern that emerged is that the majority is concentrated in the first three requirement pairs and is almost evenly distributed over only three types of communication: Communication of changes, Coordination of activities and Requirements clarification.

When these findings are considered in the context of requirements engineering, they can be explained by the nature of interaction inherent in these communication types. Most project members are involved in communicating changes, coordinating activities, and clarifying requirements on a daily basis. In contrast, the activities of Requirements negotiations may be less frequent and also only involve certain members. ORGs guidelines require that team leaders negotiate requirements with business partners, and thus requirements negotiations should happen outside the development team and brokerage happens with the business liaison.

Further, the five networks without brokers are not equally distributed but mostly located with dependency D_4. This finding can be explained by the fact that all team members in D_4 are part of both teams and I studied only brokerage in which at least one member is in only one requirement network.

Does distance really matter? Perhaps the most interesting and surprising pattern is that the most frequently identified broker in all configurations is located in the US (AC, development leader). One would not be surprised that *AC* mostly communicated across distances given that ten out of fourteen project members are in Brazil, and because she played a key role in the project, as a development leader. Given that geographical distribution introduces significant communication problems [68], and that maintaining relationships across distances is known to be difficult [48], one would expect that for more efficient communication she would have appointed or collaborated with a Brazilian-based project member or leader.

Surprisingly, that was not the case. *AC* was not only communicating actively with the distanced members but was a broker for all communication types among Brazilians. I would have expected to see this pattern with a Brazilian-based project member. Particularly interesting is that she was a consultant in two cases, despite the

distance from Brazilian members. This can be explained by the familiarity of members with *AC* given her role, and corroborates with prior evidence that familiarity has a positive influence on communication patterns [48]. Other factors that could explain this effect are knowledge and experience in the project, as discussed next.

This finding implies that organizations planning to establish a remote team with requirements-driven technical dependencies could mitigate the effect of distance by ensuring that the remote team includes experienced team members.

Knowledge and experience as determinants for brokerage. Additional contextual project information reveals other factors that relate to brokerage. Allocated to three out of project eighteen requirements, *AC* was part of four of the five pairs of dependent requirements. I believe that her knowledge and experience is a strong determinant for her broker role in most networks. *AC* has been a development leader at ORG for more than seven years. She not only acquired extensive knowledge of the project in her role as coordinator of negotiation activities with the business partners, but ORG also has the practice of copying development leaders on all project-related email communication. Her profile fits what has been referred to as a specialist role and leads her to become a broker in the team's communication. Her ability to act as a broker of communication with the distanced members is an example of a group structure in which active communication counteracts the effect of knowledge pockets in organizations.

Brokers as gatekeepers of information. When particular configurations of brokerage were considered, special types of brokers emerged that mediated the flow between interdependent networks. This corroborates with the literature on the role of gatekeepers in coordination processes of interdependent work teams [44] [67].

In the context of requirements engineering, brokers who act as information flow agents between two dependent requirements networks emerge as gatekeepers in processes of requirements change management. For example, a broker from the dependee network that mediates outgoing information to the dependent network is in a position to control the flow between those who need to send the change information to those who need to receive it because their work is affected by the change and need to effectively coordinate. Similarly, brokers from dependent network that mediate incoming information into the dependent network will be in a position to control the receiving of change information and thus affect the ways in which those in the dependent

network can react to change.

Two interesting patterns of outgoing and incoming brokerage emerged. First, as expected, most brokers of information flow are those members who are related to both requirements. On the one hand these people have most knowledge about people and their work in both networks. In the context of requirements management processes, these brokers would be in a better position to know who in the dependent network needs to receive the change information when sent from the dependee network. On the other hand, in the SHIP project most project members in the RCSNs were related to both requirements. This results in a higher chance that brokers are located in the intersection of two requirement networks.

Second, one would expect that most frequent configurations that needed brokerage would be those in which the sender and receiver of information work in different requirements and the broker would be located at the intersection of the two networks. However, the most frequently observed brokerage was for the communication between a member in only one requirement and a member working on both requirements (e.g., brokers b_5 and b_8 in Table 6.16). This is surprising, given that in all these cases the members' work assignment to the same requirements would imply direct communication not mediated through brokers.

Table 8.4: Summary of findings about information flow within RCSNs by project

Finding	**SHIP**	**APP**
RCSN is disconnected	Little	Largely
Members can be reached	Most of them	Few
Members that cannot be reached	Emergent	Developers
Info flow may break if a person is removed	Yes	Yes
Info flow disruption may be caused by ...	US dev lead	Testers
Most communicated with others	Dev lead	Tester
Most received information from others	Leaders	Leaders
Member more aware of others	BR Dev lead	Tester
Member who others are most aware of	Dev leads	Tester
Brokers are present	Yes	–

8.5 Alignment among Coordination Needs, Actual Coordination, and Organization Structure in RDC

Research question 5 asked *"To what are extent requirements-driven collaboration patterns aligned with coordination needs informed by requirements dependencies? Moreover, to what extent do requirements-driven collaboration patterns follow the organizational structure?"*. This question has been answered by empirically examining the communication-RCSNs (called AC, short for actual communication) and the coordination needs (CN) established by the requirements dependencies in each project. I used the socio-technical congruence measure proposed by Cataldo as well as the Role-based measure proposed in this research to analyze the networks. The Role-based measure extends Cataldo's socio-technical congruence measure in the sense that communication channels imposed through the organizational structure are taken into account to describe coordination behavior using a requirements lens. Results of the Role-based measure revealed patterns of collaboration in a more finer-grain detail than Cataldo' socio-technical congruence measure can bring to light. Below I discuss each of these patterns.

Backchannel communication does take place in requirements-driven collaboration. Backchannel communication (BCc) is information that flows in misalignment with the prescribed organizational structure. Usually team members establish "behind-the-scenes" communication to overcome obstacles imposed by the organization to the movement of information or to faster obtain information necessary to accomplish their tasks [55]. I found that backchannel communication does take place in requirements-driven collaboration (see *Actual* numbers in the third cluster of columns in Figure 7.7 and Figure 6.7), and that it happens in different degrees. For instance, results showed that in APP over four-fifths of the reported actual communication is backchannel communication. In contrast, results revealed that backchannel communication in SHIP represents less than one-fifth of the actual communication.

The amount of backchannel communication is an indication of the extent that coordination activities are aligned with the organizational structure. A high amount may be a symptom of an organization structure that does not support collaboration among team members throughout the project life-cycle, while a low amount may

attest that actual coordination is in line with the management vision of how collaboration should take place. However, the few backchannel instances suggest that there is still need for members to contact colleagues playing roles they should not have direct access to.

In APP, senior management split the team into four small groups by business area that each set of applications attends and defined an organization structure centralized on the leaders of each team in order to maintain control and to minimize the impact of the team's lack of expertise on the applications. Maintenance ownership of APP portfolio had recently been transfered to Brazil, and senior management wanted to ensure that the headquarter was going to be satisfied with the quality of work offered by the Brazilian team. The high amount of backchannel communication identified by the Role-based measure reveals that requirements-driven collaboration did not take place as envisioned by APP senior management.

In contrast, the low amount of backchannel communication reflects SHIP's decentralized organization structure where developers and testers can directly coordinate with each other. Three-quarters of these communication refers to 2 (out of 5) sets of dependencies that have in common the requirement $SHIP{-}R_5$. This suggests that this requirement was problematic and that it affected the team's coordination abilities, leading members to disrespect the organization structure to more effectively coordinate work related to requirement $SHIP{-}R_5$.

Test team is predominant in initiating backchannel communication. Testing is an important activity in software development, which provides an independent evaluation of the quality of the software and assesses whether the software meets the stakeholders needs. Although testing activities can start at any time in the development process, it is highly dependent on the definition and approval of the project requirements. Testers need to understand the requirements specifications to write and execute test cases. To ensure that testers develop this understanding without bias, current culture in software development is to separate testing from the development team. Despite any organizational separation, traditionally testers consult with requirements analysts to clarify requirements and seek support from developers to better comprehend the software architecture and then define testing strategies and test cases scenarios.

The test team is slightly predominant in initiating backchannel communication in both projects. In APP, testers only have access to outside members of their team

through the test leader which was over allocated. She was responsible not only for defining the test strategy but also for training the testers in the use of the tools adopted by ORG to execute test cases and to manage testing resources. Although the testing team followed the practice of shortly meet daily to report progress status and to share any important information related to test activities, I observed that testers needed to closely work with people who could support them in understanding the applications and the project requirements. Hence, their initiatives of seeking help from developers and requirements analysts.

In contrast, SHIP' structure allows testers to directly coordinate with developers. Although this flexibility in relation to APP, most of the backchannel communication were initiated by the test leader and targeted developers, and were about the sets of dependencies that have the problematic requirement $SHIP{-}R_5$ in common. I observed that the test leader was very active during the technical group meetings with the development team or the management meetings with the project manager, development leaders, and business representatives. She frequently summarized the action plans and questioned colleagues about decisions made. By asking them to explicitly express the rational followed to made decisions she frequently identified that members of her team were not clear about the requirements or that there were misunderstandings about which data set to use to execute test cases. This strong behavior and sense of responsibility probably led her to seek development teammates to help her team to get work done. In addition, although she has been working in the APP team for about 2 years she was new to the leadership position. This was her first release as test leader and she was receiving intense support from the former test leader, who was also Brazilian, and from her testing manager in US, also a former test leader for APP. The responsibilities that come along with the leadership of a critical application of ORG's shipping business process likely reinforced her will to ensure that her team had all the information it needed to work. Even if it meant sometimes disregard the organization structure to acquire the needed information.

Team dispersion did not stop members of establishing backchannel communication. Results in Table 7.18 show that about one-third of APP's backchannel communication was initiated by the test team, and took place with outside team members. The test team was physically separated from requirements analysts and developers. This isolation was designed on purpose to avoid bias from others when defining test strategies and test cases scenarios. Despite the short distance, testers

walked about about 150 meters (or about 5-10 minutes), walked two floors up, and went through a strict security checking protocol to reach their colleagues in the neighbor building to communicate face-to-face. Past research (e.g., [8], [84], [77]) has shown that 30 meters of physical distance is enough to reduce daily contact, awareness of colleagues, and most important, frequency of informal communication, thus decreasing collaboration activity. In addition to being distant enough from each other, testers were new hires and contractors. Hence, the test team had no previous knowledge about their colleagues or working practices at ORG. Despite this lack of awareness, they were informed that developers conducted a in-depth reverse engineering activity to learn about the applications, and that requirements analysts were the people responsible for specifying and managing the requirements. The need for information potentially drove the test team to overcome the short but disruptive physical distance and establish informal communication, contradicting previous research.

Distance also did not stop SHIP's team members of establishing backchannel communication. Most of the backchannel requirements negotiation (RN) and notifications of changes communication (CC) was initiated by team members located in the US headquarter office and targeted members placed in Brazil. Since SHIP team was working together for approximately 3 years and adopt the practice of sending newcomers to work for a while with people in the US headquarters, all members have had the chance to met face-to-face at least once. These face-to-face working sessions helped members to build trust on each other and integrated the team in a way that language and culture barriers were minimized. Therefore, whenever an American team member needed to coordinate work with a Brazilian (or vice-versa) he did not hesitate in contacting his remote colleague and went ahead to avoid down time or rework.

In maintenance projects, technical information about the product is relevant during the negotiation of requirements phase and contributes to the definition of the project scope. Despite APP's process definition that requirements must be negotiated among business partners, project managers, and requirements analysts, and that development leaders and the test leader can be invited when necessary, results in Table 7.18(a) reveal that developers got involved in deciding the project scope. Since the team was new to the product, during the requirements negotiation requirements analysts invited developers to provided technical information that would help them and the project managers to estimate the time and the effort

necessary to implement some requirements. The scope of APP releases consists of as many requirements as it is possible to implement and deliver within a certain time window. Thus, requirements analysts counted on the expertise that developers acquired during the reverse engineering activity to define the effort and time necessary to implement certain requirements. New software development may not follow the same behavior since there is no previous system to have knowledge about.

APP project managers were surprised when they learned that requirements analysts were seeking help to decide whether to include a requirement on the project scope from developers instead of from the development leaders. Because each development leader supervised the reverse engineering activity for its own subgroup, the managers assumed they have developed expertise on the applications in a deeper level. The Role-based measure revealed that these interactions were not in accordance with the organization structure.

Team members go beyond the organizational structure to seek for requirements clarification. Usually when one needs help, one needs it right away [34] because the information sought is critical for one to complete his work. The high amount of backchannel communication in APP is an indication that members could not wait to obtain requirements clarifications thus they chose to ignore the organizational structure to timely acquire what they needed to continue performing their work. The majority of these "behind-the-scene" communication in Table 7.18(b) involved a developer. Because of the reverse engineering strategy adopted by senior management to the Brazilian team acquire knowledge about the applications and their architectures, developers became the experts on the applications. Thus, it is not surprising that they were frequently sought for their requirements analyst and tester teammates to clarify requirements. Note that the test team went to the requirements analysts to get support to understand the requirements and then be able to more effectively write test cases. The results corroborate Ehrlich and Chang [48] finding that when it came to acquiring information, members of these teams were more likely to seek specific technical information or administrative help from people outside their software team. In addition, the Role-based measure reveals not only that developers are being sought by colleagues from other teams but also that these clarifications characterize backchannel communication.

Although in SHIP the proportion of backchannel communication regarding requirements clarification is lower than in APP, this is the highest amount of backchan-

nel communication in SHIP. The Role-based measure discloses that most of the backchannel communication in Table 6.19(b) was initiated by the test team and has senior development team members as target. These senior development members are internally called "core-team" by their teammates because of the in-depth knowledge they have about how the application works and interfaces with other critical ORG applications. For instance, during the three months on site I constantly observed testers using instant messenger to call a senior US developer to help them understand the protocol used to call a component responsible to connect all manufacturing plants together. Some of the requirements that the testers were implementing were related to update the way that the plants communicate to each other to reduce shipping time and, as a consequence, to improve customer experience. I also joined testers and developers in long lunch breaks where testers took the opportunity to discuss the project requirements and their technical-related issues. Note that lunch can take as long as 2 hours, and it is usually a good time to socialize since Brazilians like to go out to have lunch at the same time as a group [16]. It is likely that testers were referring to these informal situations when reporting communication with developers.

Team members overlook the organizational structure to timely notify their teammates of changes made to requirements and technical specifications. Changes to requirements and design specifications need to be communicated promptly to team members to avoid negative impacts on quality and team productivity or need to rework. In APP the concern for having the teammates aware at the appropriate time of changes to requirements may have urged the requirements analysts to overlook the organizational structure and directly communicate developers and testers of approved changes. Aware of how critical it was for the office in Brazil to demonstrate its ability to perform well, requirements analysts mentioned in interviews that they opted for informing directly their developer mates (as it can be seen in Table 7.18(c)) instead of following the organization structure and communicating development leaders because they had the perception that notification of changes could get lost in their colleagues' overloaded mailboxes.

Also, because the test team writes the test cases early in the development phase as soon as they receive the requirements specifications, it is common that testers identify opportunities for improving technical specifications and communicate them to developers. In follow-up interviews I identified that the backchannel communication between the test and the development teams were about these improvements, which

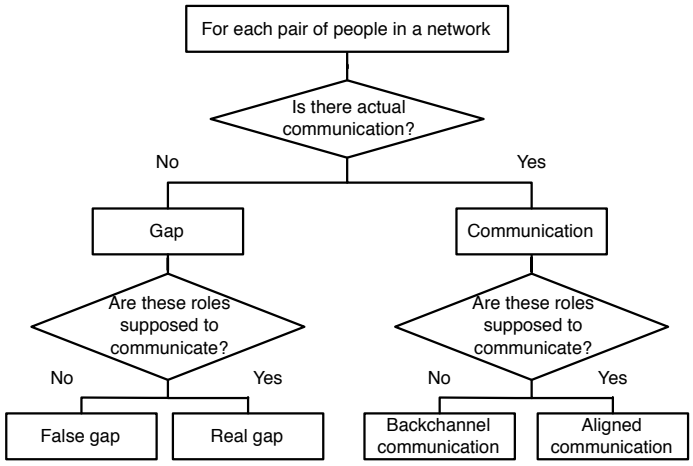


Figure 8.1: Flow chart to classify coordination activities

resulted in changes to technical specifications that track back to some requirements in the set of dependencies that we investigated.

In addition to the identified patterns, the information revealed by the Role-based measure can be organized in a model that categorizes coordination activities. The model in Figure D.2 can be applied to coordination in the presence of an imposed organizational structure among roles. For each pair of people in a requirement-centric social network, one can examine the relationship between them and follow the chart to determine the type of coordination activity that connects this pair.

This preliminary model reveals four different types of relationships between people in a project that allow the **classification of coordination activities** among team members. These relationships are as follows: *False gap*, where there is no actual coordination but roles were not supposed to talk to each other; *Real gap*, where there is no actual coordination but coordination needs exist; *Backchannel communication*, where there is actual communication between roles who are not supposed to talk to each other; and *Aligned communication*, where there is actual communication and this communication satisfies coordination needs.

Also, the results of the empirical investigation of the alignment between coordination needs and the organization structure suggest that false gaps and backchannel communication may be indicators of whether an organizational restructuring may lead to a better alignment of social and technical coordination. Thus, the Role-based measure can be used as a tool to **assess the health of the organization structure**. I propose the scheme presented in Table D.2 to support this assessment.

Table 8.5: Scheme proposed to assess the organization structure's health

	Few false gaps	Many false gaps
Few backchannel communications	**Nominal situation (1):** Organizational structure aligns with technical dependencies and people are following prescribed communication channels.	**Possible problem situation (4):** people may not be communicating enough.
Many backchannel communications	**Possible problem situation (2):** people are breaking communication structure arbitrarily despite alignment of communication structure with technical dependencies.	**Possible problem situation (3):** poor organizational structure is causing people to compensate using back-channel communication.

The best-case scenario (1) is if there are a small number of false gaps as well as a low amount of backchannel communication. In the empirical study, the SHIP project falls in this category. This nominal situation means that there is good alignment between the project's technical dependencies and role responsibilities, and that people are obeying the organizational structure; it may also suggest that backchannel communication is not needed due to the good alignment between a project's technical dependencies and organizational structure.

A situation that is slightly worse (2) is if there are few false gaps but a lot of backchannel communication—this situation implies that the individuals are not respecting the communication structure despite the fact that the organizational structure is well-aligned. Individuals who are found to constantly break the guidelines may need to be disciplined.

If there is a large number of false gaps, as well as a high amount of backchannel communication (3), this implies that the organizational structure is not working for reasons such as delays in propagating information through the formal channels [57], but that people are compensating through their own means. This is the case of APP project in the empirical study. In this situation, the organizational structure is working against people and thus should be modified.

If there is a large number of false gaps, but low backchannel communication (4), then more information about the organization is needed. It would appear that false gaps would get in the way of individual productivity, but there are two reasons that

backchannel communication may not actually occur: there is actually no need for it; or individuals are adhering to the communication structure to the detriment of the project.

The identification of the organizational structure's health in light of the coordination behavior has to be assessed with support of contextual information. Managers need to consider the influence and impact of organizational processes, policies, and guidelines to decide whether changes on the organizational structure are necessary. The Role-base socio-technical congruence measure only provides the means to inform the managers about detailed information on coordination behavior and its alignment with the organizational structure.

Table 8.6 summarizes the findings about alignment between coordination needs and actual coordination, and coordination needs and organization structure by project.

Table 8.6: Summary of findings about RCSNs alignment by project

Finding	**SHIP**	**APP**
Extent of alignment between CN and AC	Mid	Low
Extent of alignment between CN and org structure	High	Low
Amount of misleading gaps	Few	Many
Amount of backchannel communication	Few	Many
Test team initiates backchannel communication	Yes	Yes
Distance did not stop BCc of taking place	Yes	Yes
Previous application knowledge was relevant for RN	-	Yes
Members seek for clarifications beyond org structure	Yes	Yes
Team members overlook org structure to CC	-	Yes

Chapter 9

Final Considerations

This chapter presents the research validation strategy adopted throughout the research process, the contributions and implications of the research findings for the requirements engineering community, and some directions to future work.

9.1 Research Validation

I relied in three criteria proposed by Strauss and Corbin [126] to validate the quality (innovation, usefulness) and the credibility (trustworthy, reflection of the phenomenon characteristics) of the contributions of my study: *fit*, whether results resonate with the audience (researchers and/or practitioners) for whom the research was intended to; *applicability or usefulness*, whether findings offer new insights about the investigated phenomenon; and *sensitivity*, whether data and findings were derived from research questions or research questions were posed inspired on the analysis of data collected. Throughout the research process I adopted the following strategies suggested by Creswell [32] and reinforced in the software engineering field by Easterbrook et al. [47] to validate the three criteria. For each strategy, I provide a brief definition and description of the actions I took to put the strategy in place.

Triangulation. *The use of different sources of data to confirm results and build a coherent outcome.* To answer each research question, I collected data using multiple data collection methods and data sources. Data were triangulated before starting the analysis process aiming to assure its accuracy. Mainly, interviews were used to confirm data collected through document inspection and observations. For instance, there were small discrepancies about requirements dependencies among requirements

documents, project planning spreadsheets, and testing documentation. Thus, I conducted a series of interviews with development and test leaders as well as project managers in order to gather these dependencies. I then compared the two sets and conducted a few additional interviews to establish the final set of dependencies, which I used in my investigation. Also, communication data filled out in the work diaries by most of the team members were compared against communication data reported on the questionnaire. There was a second researcher on site. Although his research had a different goal than mine, most of the time we observed the same team members working at the same time. This allowed me to compare my notes with his seeking to identify if the actions we observed were similarly perceived.

Member checking. *When the researcher goes back to the participants to ensure that the interpretations of the data make sense from their perspective.* The three months I spent on site gave me the opportunity to consult with team members about data collected. Most of my inquiries regarded the understanding of the requirements specification and dependencies, as well as the task allocation plans. This information was used to construct the assignment networks, which worked as baselines for customizing the questionnaires and for asking about communication interactions. Hence, my concern in assuring that I understood and used the data properly.

Since the scope of the APP project consisted of enhancement requests for over one-hundred applications and the team was new to them, sometimes I collected incomplete data about requirements during observation sessions. I adopted the protocol of periodically interviewing team leaders or senior developers to complete or clarify the data. They would then provide me additional documents or show me the information on their computer screen and patiently explain it to me. These clarification interviews were voice recorded in order to allow me to listen to the explanation later on again if necessary. I also followed this clarification process in SHIP but I did it with less frequency because the requirements information obtained per observation session were mostly complete for this project.

Interview to confirm findings accuracy and usefulness. *When the researcher interviews participants of the case study for accuracy and usefulness of the results.* Over the years I had the chance to return several times to ORG in Brazil to discuss partial results of my investigation. In addition to interviews, I presented the results to the team members of both projects and to senior management. These presentations

were organized in two parts. First, I would present the results and answer clarification questions. Second, I would ask the participants to explain why certain collaboration patterns were found and use their insights to corroborate with my explanations and interpretation of the results. For example, in one of these presentations a member of the SHIP project suggested an alternative explanation about why little interaction was observed with the project system architect. He explained that despite the fact that the system architect was allocated to work on SHIP, he was also allocated to a new release of the Shipping system which was migrating the application to a new technology. Thus, most of his activities were concentrated on defining a new architecture for the Shipping system instead of maintaining the old architecture. He also explained that the Brazilian development leader acted as a backup of the system architecture, performing some of his tasks. This contextual information was missed during the data collection process but thank to the presentations I had the chance to correct it in time to change my interpretation of the results. This review and discussion sessions also worked as an indirected way of validating the observations and interviews conducted earlier on.

These presentations were valuable not only to confirm the accuracy of the findings but also to discuss their usefulness. Senior managers and project managers of the APP project were surprised to learn that most of the communication reported by APP's members took place between roles that should not have communicated according to the structure they have defined for APP. This structure was intentionally centralized on the project leaders aiming to increase control and ensure that leaders were aware of what was going on in the project. Team members mainly ignored this structure claiming that they could not afford waiting for the leaders to propagate information and that it was easier to discuss the matter with the final target since they were collocated than risk adding noise to the conversation by going through the leaders. Team members were satisfied to hear from me that the team apparently functioned in practice despite the misalignment with the organization structure.

I annually conducted short interview sessions with representatives of each project and role to collect their opinion about which aspects of collaboration they would be interested on. For example, one APP manager asked me if it was possible to tell him how communicative the new hires and contractors were and how aware they were from the remaining of the team. He was interested in finding whether he had to take any action to better integrate the novices with the senior members to speed up productivity. My first thought was to run a degree centrality test in each requirements-centric

social network to identify the level of activity of each member per network. However, I added to the degree centrality results the core-periphery test which indicates which members are closely connected others. Thus, I also reported on the composition of the network cores in terms of the participation of the novice members on the cores. I avoided discussing specifics in individual interviews with managers to protect the members privacy and ensure that my findings were not being used to assess the members' performance. This request from one of the APP managers led me to learn about a new social network measure (core-periphery test), and as a consequence, I extended the set of measure proposed in the framework.

The chance to periodically visit and discuss the research and results with team members, and to speak freely with them is rare in the Software Engineering community. Team members are usually busy or are allocated to work on other projects. The constant feedback I received from ORG team members over the three subsequent years of the data collection was a valuable mechanism to assure the validity of the contributions.

Rich, thick descriptions. *The use of detailed descriptions to convey the setting and findings of the research.* Since I visited the team members on their work environment and stayed on site for three months, I had the chance to observe how the project teams collaborated in practice and to interview team members about project characteristics. Data collected through these two data collection techniques provided rich contextual information for the understanding of how requirements-driven collaboration took place in the two investigated projects. This rich contextual information was substantial in the interpretation of the patterns found when analyzing the requirements-centric social networks. The in-depth knowledge about the team members and their work style helped me to drawn conclusions about the results found in my investigation.

Clarify bias. *The use of honesty with respect to the biases brought by the researchers to the study, and the use of this self-reflection when reporting findings.* I am a former ORG employee, but although my past experience at ORG I have never directly worked with any of the research participants. Four years prior to starting the case studies investigation, I conducted a large research project which proposed a major organization change. It is likely that this successful previous research project influenced the decision of the senior managers in welcoming my research colleague and I to con-

duct a new research project at ORG. This previous research experience also created expectations that ORG would directly benefit from the research results. These expectations may explain why ORG team members were very open in receiving feedback and annually discussing findings with me.

I understand that my previous knowledge about the company and its processes facilitated the data collection process and the interpretation of results. Thus, I believe that this previous knowledge positively affected my research. To avoid jeopardizing my analysis, I was careful to not draw conclusions based solely on my previous experience as an employee. I took the practical action of verifying the influence of potential bias. I conducted a reliability test [140] of a sample of the findings with the second researcher present on site. I provided him with the plot of some random requirements-centric social network for both projects and asked him to report his interpretation of the reading of the sociograms. Next, I asked him to explain to me why he wrote down each result, and compared his findings with mine. I also applied some social network measures, such as brokerage, and asked him to explain why certain results were found. Once again I compared his explanations with mine. Although we did not agree one-hundred percentage of the time, we discussed the discrepancies together aiming to identify why we have interpreted the facts in divergent ways. These interpretation sessions took place over time and were intensified during periods where I was writing conference or workshop papers. All discrepancies were solved during these sessions.

Prolonged contact with participants. *When the researcher makes sure that exposure to the subject population is long enough to ensure a reasonable understanding of the issues and phenomenon under study.* My main visit on site lasted three months, which coincided with the beginning and closing phases of both projects. Not only the length of time visiting the team is longer than the average in Software Engineering research projects (e.g., Sharp and Robinson study lasted one week [118]) but also the relationship developed with the team members was deeper than traditionally expected. I was welcomed to observe the members for the entire work day, to attend all the meetings scheduled, and to join them at lunch time or afternoon coffee breaks. I spent three months walking around, from cubicle to cubicle, with my paper notebook and voice recorder without having any request of interaction rejected or having heard a complaint that I was disturbing the work environment. I was sensible enough to perceive when tension was in the air and excused myself without being asked to leave.

I periodically met with senior managers who were concerned in finding out whether the team members were being polite and giving me enough attention. The company is used to researchers visiting the Brazilian unit since ORG is located on a university campus and the company has a research agreement with the hosting university that was almost ten-years old at the time of the main data collection period.

Despite the fact that my visit was limited to the Brazilian site, I was welcomed to call and exchange e-mail messages with the team members located in the United States and India who accepted my invitation to participate on the research. The company even provided me with an access code to call the remote members. To compensate from not observing the distributed members in person, I interviewed them to profile their work preferences and learn about their work environment. The long stay and in-depth established relationships allowed me to understand the project context, to inspect documentation and to customize the questionnaire while on site, and to deliver the questionnaire and to face-to-face answer the respondents requests for clarifications.

Peer debriefing. *When a researcher has a peer debriefer to whom he can ask questions about the study and the assumptions present in the reporting of it, so that the final account is as valid as possible.* In addition to my supervisor, I discussed my findings with the second researcher on site as mentioned on the *Clarify bias* strategy item. Another person I discussed the results with was a researcher partner who is a professor associated to the university where ORG has its development center unit. This professor has been working on research projects in partnership with ORG and with my research laboratory for over five years. Discussions with this research partner allowed me to interpret the results from different angles. He also called my attention to important aspects that I was exploring in-depth in my discussion of the results.

External auditor. *The same as peer debriefing, except instead of using a person known to the researcher, an external auditor to review the research procedure and findings is used.* In order to have the research results reviewed by external auditors, I submitted papers to referred international conferences and workshops, as well as I wrote chapters for books reviewed by formal committees (see Section 9.2.1). Reviewers criticisms motivated me to self-reflect about the findings and aspects of collaboration that were under investigation. The majority of the reviewers' suggestions of improvements were implemented and incorporated in this dissertation document.

9.2 Contributions

This dissertation brings several theoretical and empirical contributions to the areas of Requirements Engineering. In terms of theoretical contributions, this dissertation presented a **framework** for investigating requirements-driven collaboration. This framework consists of an approach based on a social structure that focuses on a requirement as the unit of work around which collaboration occurs. This structure represents a cross-functional team whose members' work activities are related to one or more interrelated requirements, as well as downstream artifacts. The framework also outlines a number of social network analysis measures to study collaboration aspects driven by the development of requirements and the team members social interactions. These measures are applied on the requirements-centric social networks which represent the collaboration behavior in requirements-centric teams. The proposed framework for studying requirements-driven collaboration provides the necessary conceptual instruments to examine patterns of communication and fleeting knowledge in software teams. It provides sufficient flexibility to be extended in terms of additional social network analysis measures and of requirements-driven collaboration aspects.

The second theoretical contribution is the proposed **Role-based socio-technical congruence measure**. Based on the current socio-technical congruence measure defined by Cataldo et al. [24], the Role-based measure aims at examining requirements-driven collaboration in the presence of a formal organization structure that prescribes communication channels in the organization. In other words, the proposed Role-based measure indicates the extent that actual coordination attends coordination needs that are aligned to the formal imposed communication channels.

The empirical application of this extended measure showed that the proposed measure has more explanatory power of coordination behavior with respect to influence of roles on amount and type of communication than Cataldo's original measure. A missing communication between any pair of members is considered a congruence gap by Cataldo's measure. In the Role-based measure, congruence gaps between members who play roles that should not directly communicate according to the organization structure are distinguished from the other gaps. These gaps are named **false congruence gaps**, or *false gaps* in short. The identification of false gaps is important because they highlight point-to-point direct coordination needs that might not have to be satisfied. The proposed measure also allows one to identify **backchannel communication**, which is communication that takes place between a pair of members

who play roles that should not directly communicate with each other. In addition, backchannel communication properties are also identified, such as who are the roles involved and what topics team members discuss when communicating "behind-the-scenes". Managers can benefit from the awareness that backchannel communication is happening in their projects, and use the detailed information about this type of communication to align or to supplement the formal organization structure.

In addition, the information revealed by the Role-based measure can be organized in a model that categorizes coordination activities. For each pair of people in a requirement-centric social network, one can examine the relationship between them and follow the chart to determine the type of coordination activity that connects this pair. This preliminary model reveals four different types of relationships between people in a project that allow the **classification of coordination activities** among team members. These relationships are as follows: *False gap*, where there is no actual coordination but roles were not supposed to talk to each other; *Real gap*, where there is no actual coordination but coordination needs exist; *Backchannel communication*, where there is actual communication between roles who are not supposed to talk to each other; and *Aligned communication*, where there is actual communication and this communication satisfies coordination needs.

Furthermore, results of the empirical investigation of the alignment between coordination needs and the organization structure suggest that false gaps and backchannel communication may be indicators of whether an organizational restructuring may lead to a better alignment of social and technical coordination. Thus, the Role-based measure can be used as a tool to **assess the health of the organization structure**. The identification of the organizational structure's health in light of the coordination behavior has to be assessed with support of contextual information. Managers need to consider the influence and impact of organizational processes, policies, and guidelines to decide whether changes on the organizational structure are necessary. The Role-base socio-technical congruence measure only provides the means to inform the managers about detailed information on coordination behavior and its alignment with the organizational structure.

This dissertation also brings significant empirical contributions. A substantial initial body of knowledge about requirements-driven collaboration is developed grounded in empirical evidence. Although only two projects were investigated, this multiple case study research revealed that **requirements-centric social networks are dynamic and involve considerable cross-sites and cross-teams interactions** despite

of how new or experienced the team is. Team members exchanged information to support the development of requirements-related activities with people who are not assigned to work on the requirements, named **emergent members** but somehow had knowledge to contribute with their colleagues. Members also sought emergent members for help, engaging more people in the project than initially estimated.

This research also identified some interesting differences in the coordination behavior of novice vs. experienced teams. The novice requiremens-centric social networks in the team were **less fully decentralized** than in the experienced teams, meaning that collaboration was more dependent on key people who are critical for the development of the project requirements than if the structure was fully decentralized. The more experienced team, which has an in-depth knowledge about how the product functions and is aware of how new requirements will impact the existing application, formed very **decentralized network structures** where team members collaborated more equally with each other. When requirements clarification is the topic of discussion, team members tend to seek help from colleagues of the same team thus forming tightly-coupled groups. However, when propagating notifications of changes, team members communicate with others outside their group aiming to inform their colleagues in a timely manner and avoid rework.

In addition, findings unveiled that there are **key members, namely brokers, controlling the information flow** in requirements-centric social network of dependent requirements. The power of controlling this flow lies within the hands of only a few members in the project. These members were identified as experts on the application for working over a long period of time in the project.

The application of the Role-based socio-technical congruence measure in the two investigated projects disclosed that **backchannel communication does take place in requirements-driven collaboration** meaning that team members go beyond the organizational structure to coordinate requirements-related work. Members of the test team are predominant in initiating backchannel communication. This finding suggests that testers should be more engaged in requirements-related activities in order to facilitate their work.

9.2.1 Published Papers

In support of the contributions of my research, the following papers have been published.

Book Chapters

Damian, D., Kwan, I., Marczak, S., **Requirements-Driven Collaboration: Leveraging the Invisible Relationships between Requirements and People**, Collaborative Software Engineering, Mistrik, I., Grundy, J., van der Hoek, A, Whitehead, J. (Eds.), Chapter 3, pages 57-76, Springer-Verlag, Computer Science Editorial Series, London, England, March 2010.

Damian, D., Marczak, S., Kwan, I., **Requirements Engineering in Global Teams**, Global Software and IT: A Guide to Distributed Development, Rightshoring and Outsourcing, Ebert, C. (Ed.), IEEE Computer Society Press, Wiley, Los Alamitos, USA, 2011. (To appear)

Peer Reviewed Conference Papers

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

Marczak, S., Damian, D., Stege, U., Schröter, A., "Information Brokers in Requirements-Dependency Social Networks", in *Proceedings of the International Requirements Engineering Conference*, Barcelona, Spain, pp. 53-62, IEEE Computer Society, September 2008.

Peer Reviewed Workshop Papers

Kwan, I., Damian, D., Marczak, S., "The Effects of Distance, Experience, and Communication Structure on Requirements Awareness in Two Distributed Industrial Software Projects", in *Proceedings of the Global Requirements Engineering Workshop, in conjunction with the International Conference on Global Software Engineering*, Munich, Germany, pp. 29-35, IEEE Computer Society, August 2007.

Marczak, S., Damian, D., Stege, U., Schröter, A., Kwan, I., "Patterns of Infor-

mation Flow in Interdependent-Requirements Social Networks and Implications for Requirements-Driven Collaboration", in *Proceedings of the Socio-Technical Congruence Workshop, in conjunction with the International Conference on Software Engineering*, Leipzig, Germany, (available online), IEEE Computer Society, May 2008.

Marczak, S., Kwan, I., Damian, D., "Investigating Collaboration Driven by Requirements in Cross-Functional Teams", in *Proceedings of the Collaboration and Intercultural Issues on Requirements Communication, Understanding, and Softskills Workshop, in conjunction with the International Requirements Engineering Conference*, Atlanta, United States, (available online), IEEE Computer Society, September 2009.

Peer Reviewed Extended Abstract and Poster Papers

Marczak, S., Kwan, I., Damian, D., "Social Networks in the Study of Collaboration in Global Software Teams", in *Proceedings of the International Conference on Global Software Engineering*, Munich, Germany, IEEE Computer Society, August 2007.

Kwan, I., Marczak, S., Damian, D., "Viewing Project Collaborators Who Work on Interrelated Requirements", in *Proceedings of the International Requirements Engineering Conference*, New Delhi, India, pp. 369-370, IEEE Computer Society, October 2007.

Technical Report

Marczak, S., Stege, U., "Graph Theory Applied to Social Networks on Requirements Engineering: An Investigation Based on an Industrial Case Study Conducted in Brazil", Department of Computer Science, University of Victoria, 19 pages, TR no. DCS-320-IR, December 2007.

9.3 Implications

This dissertation also brings implications for researchers, tool designers, and practitioners. A researcher using the proposed framework for studying requirements-driven collaboration can define network relationships and actor attributes that apply to a

specific case of interest and examine different patterns of requirements-driven collaboration. Tool designers can use the proposed framework to develop tools to support requirements-centric teams coordinate effectively. Practitioners can gain insights about collaboration behavior, communication and fleeting knowledge processes, and alignment of actual coordination with the organization structure by applying the proposed framework in their organizations.

9.3.1 Implication for Researchers

Researchers have available now a framework that can be used as a mechanism to learn fine-grained details about requirements-driven collaboration. Using a requirement as the unit of analysis, this framework proposes the investigation of collaboration of requirements-centric teams composed of cross-functional members. This approach allows the study of collaboration that spans the entire project life-cycle and the examination of collaboration beyond the relationships established among developers. The fine-grained requirements-driven collaboration patterns identified by using the framework in real case situations can bring insights to researchers of how to propose better tools and processes to support cross-functional teams' collaboration.

To extend the study of requirements-driven collaboration, researchers would have to examine different types of cases of requirements-driven collaboration. I investigated two maintenance software projects with sets of two functional dependent requirements. The framework can be applied to larger projects that contain multiple dependent requirements. This will provide insight into the nature of requirements-driven collaboration over complex technical dependencies. Moreover, the framework can be applied to new software development to bring insights into requirements-driven collaboration patterns when the requirements and the application are new to the team. In addition, non-functional requirements can be examined in order to broaden the current knowledge developed in this dissertation about requirements-driven collaboration. Non-functional requirements may not have clear boundaries, thus it may be not as straight-forward to identify who is assigned to work on them, potentially making necessary to extend the proposal of how to construct the baseline requirements-centric social networks, named *assignment-RCSNs*.

The study of requirements-driven collaboration can also include in addition to project documentation and survey data the analysis of digital artifacts. These artifacts may include e-mail data, requirement databases, and source code that are related

to a requirement. This analysis will contribute to the understanding of collaboration driven by requirements and the traceability to their downstream artifacts.

The effect of roles on requirements-centric social network characteristics can also be studied in more detail. The two projects included interactions mainly among requirements analysts, developers, and testers, but the framework allows one to study interactions that include project managers, software architects, and any other stakeholders involved with the project as well. Also, a higher distribution of members per teams composed of diverse roles will allow the application of social network analysis measures that use the concept of group to calculate the network values, such as the E-I index measure [83].[1]

9.3.2 Implication for Tool Designers

The paradigm of a requirements-centric team can be used to develop tools to support cross-functional teams coordinate effectively. Collaborative tools can also assist managers who wish to monitor and improve coordination processes within their organization.

A tool that can generate a RCSN automatically, perhaps using data-mining techniques [12] [139] and automated requirement-traceability tools [28] can identify who works on which artifacts, and trace these artifacts to requirements. Such a tool may be able to extract data from issue-tracking repositories, requirement repositories, mailing lists, and chat logs for example. These tools can be embedded into software development and management tools to support:

- *Expertise seeking.* Generating a requirements-centric social network based on assignment and communication interactions will indicate who is assigned to working on each requirement. Thus, someone who is seeking help will be able to identify who works and communicates on a particular part of the project and consult with that person accordingly.

- *Identifying emergent requirements-centric team members.* The automatic identification of who is collaborating to the development of certain requirements at a given time contributes to the team to cope with the dynamic aspect of collaboration. As I found that more people join the requirements-centric teams

[1]The E-I index measures the extent to which the members of a group are clustered within the group structure.

as the project goes by, distributed teams can specially benefit from such a tool because these teams lack informal communication that is often used to assist in expertise location and in the identification of who should receive notifications of changes.

- *Diagnosing collaboration behavior and its evolution throughout the project life-cycle.* A tool can compute social network properties and provide a manager with information to improve project performance. Measurements include calculating the number of additional people working on a set of dependent requirements (emergent members) and identifying critical members that would disrupt the exchange of information if absent from the network (cutpoints). The tool can also identify the most active members in communicating with others. Socio-technical congruence is an example of a technique that can identify gaps in coordination. A manager may decide to increase the alignment between the social structure of the organization and the technical dependencies among artifacts. Requirements-centric social networks can also be constructed periodically to show the evolution of collaboration behavior within a period of time. These networks can be contrasted against each other and highlight changes aiming to point out differences. These differences can unveil important and critical participation (or lack of) certain team members, or indicate which are the moments in the project that most require team members to communicate with each other.

- *Broker identification.* A tool can identify brokers mediating activity on different requirements to make a project manager aware of who is intermediating communication in the project. These people may be experts on the requirements or may introduce noise to the information being exchanged. Resources can be provided to these persons so they can better do their job and avoid creating misunderstanding situations. A backup person can be trained to cover for the broker when he is not available.

9.3.3 Implication for Practitioners

Practitioners can apply this framework in real-life projects and gain insights on how to offer better conditions and improve current processes that will facilitate requirements-driven collaboration. In addition, the framework also allows the identification of the extent of alignment between actual coordination behavior and the organization struc-

ture by the application of the Role-based socio-congruence measure. The measure permits managers to identify the amount of backchannel communication, which is an indicative that people have to compensate restricted access to others because of existing coordination needs created by technical dependencies. Managers then can use the measure as a mechanism to identify the health of the organization structure. More efficient organization structures may leverage collaboration among their contributors, potentially increasing productivity and avoiding rework.

Managers can also identify which requirements are likely problematic and deserve attention by applying the framework. For instance, a communication RCSN with high density would suggest that the team members communicate a lot with each other person working on the requirement. If seeking requirements clarications is the topic of discussion in the highly dense communication-RCSN then one may conclude that the requirement is very ambiguous and problematic because it necessitates a lot of information exchange to clarify it. Similarly, a communication RCSN drawn from messages about requirements change that has high density is indicative of a requirement that is highly volatile.

In addition, the framework allows the identification of members who broker communication between two otherwise unconnected members. The presence of brokers in requirements-driven collaboration indicates that those members whose communication was brokered did not use and possible had no direct way of communicating requirements information. Misunderstandings or information loss could occur in these mediated interactions. Hence the importance of identifying whether or not brokers are present in a project. Managers can also investigate the effects of distance on communication and fleeting knowledge, gaining insights of collaborative processes in their organizations. Furthermore, all tools proposed in the previous section can be used by practitioners in their organization to examine requirements-driven collaboration patterns aiming to improve coordination processes within the organization.

9.4 Future Work

This research proposed a framework for studying requirements-driven collaboration and used the framework to empirically examine communication and fleeting knowledge in requirements-driven collaboration in two industrial distributed software projects. Below I discuss some directions for future work that I identified during this research.

Immediate work. Before this research can be extended or replicated, to avoid labor work I aim to automate the data collection and analysis processes. Social network data was collected manually. Thus, respondents were not always consistent with their answers for the fixed choice questions, where I asked the respondents to nominate any additional person up to a certain number that they had communicated with or they were aware of. In addition, collected data was manually imported to a relational database to facilitate manipulation. The manual insertion of the social data into the database was not bug-free, thus two rounds of peer review were conducted to ensure correctness of the insertion process, one by the field researcher partner and one by the research assistant. Automation of the data collection process and integration with the database solution will not only minimize the chances of inserting wrong data but will also enforce the research respondents to be consistent throughout their set of fixed choice answers.

The steps to prepare the requirements-centric social networks was automated. To construct the networks, queries were written and automatically replicated for each combination of requirements dependencies and reason of communication. Once the data to compose a requirements-centric social network was queried from the database, a script transformed the data from a relational to a matrix format. Automation saved time up to this point. Next, the networks have to be imported one-by-one to the commercial social network analysis tool used in this research (UCINet 6.0) because the tool does not allow extensions or the development of additional features. Each measure calculated required the loading of the imported network file in the commercial tool. Moreover, the visual displaying of the networks required an additional loading of the imported files. The steps of the analysis process that interacted with the commercial tool were time consuming since over a hundred combination of networks were manipulated. Thus, the repetition of the measures calculation is expected to be automated and preferably integrated with the previous steps to facilitate analysis of large data sets.

Another immediate work is already under development. This research revealed that a significant amount of collaboration takes place with team members who are not directly assigned to development tasks but somehow contribute to the project, such as business partners and former team members who act as project stakeholders. Aiming to broaden the investigation to the entire scope of people that to a certain extent contribute to the project development, a research partnership has been recently established with a researcher whose expertise is to identify social networks of project

stakeholders. Her main concern is to ensure that these networks consider each and every person involved in requesting and sponsoring the project requirements. The intended goal of the recently establish new research project is two-folded: to extend the set of social network analysis measures used to define and characterize the stakeholders social networks in order to provide a fine-grained view of the relationships established by the stakeholders over time, and to identify whether changes on the conceptualization of the requirements-centric social networks are required in order to include the entire scope of people that contributes to the project. For example, the assigned network, which is used as a baseline to compare against the actual communication and awareness networks, may have to be conceptualized in a different way.

Intermediate work. Longitudinal social networks can reveal change in collaboration behavior. Having knowledge that requirements are volatile [36] [18], I believe that the study of requirements-driven collaboration would benefit from a longitudinal study where multiple data points to construct the requirements-centric social networks over time will be established. I suggest that such longitudinal study is planned and conducted if possible.

I also would like to learn where the empirical insights hold for different project settings, such as project type, organization structure, development methodologies, tools, processes, number of requirements dependencies, among other characteristics. Therefore, I list below some suggestions for the extension of this study. I suggest to investigate:

- *larger teams* to check if the project team size affects requirements-driven collaboration behavior as suggested by literature on collaboration in software projects [15];

- *larger number of requirements on a dependency set and of dependency sets itself* to identify to what extent a larger number of requirements in a same set of dependency and a larger number of dependency sets result in different requirements-driven coordination patterns, and to learn whether the examination of RCSNs of dependent requirements would suffer of an scalability issue;

- *higher physical distribution of team members* to further knowledge of the effect of distance on requirements-driven collaboration;

- *new development projects* to identify whether the research findings hold for different project types or are applicable to maintenance projects only;
- *agile teams* to learn how the "agile way" of dealing with requirements corroborates or adds to the insights from this study. Agile teams use story cards and "the wall" as mechanisms to support coordination [119], and documentation is reduced to the minimum necessary. These different practices from traditional development are likely to have an impact on requirements-driven collaboration patterns of agile teams; and
- *non-functional requirements* to learn whether the conceptualization of requirements-centric teams and requirements-centric social networks would have to be adjusted in order to capture collaboration driven by this type of requirement.

Long-term work. The development of the proposed framework was seeded in literature review but its actual set of social network analysis measures was inspired in the empirically-informed insights obtained by the case studies. It is likely that the study of requirements-driven collaboration based in projects with different characteristics will suggest the use of additional measures or tests. Thus, the ultimate goal of extending this study as described in the intermediate work section above is to reach the framework completeness aiming to exhaust the aspects of requirements-driven collaboration that can be investigated. At this moment the unexplored aspects of requirements-driven collaboration (if any) are unknown, thus the intention of conducting more empirical case studies. The development of a theory about requirements-driven collaboration is also of interest and could be accomplished as a consequence of the further exploration of other cases and project characteristics.

I believe that tools to automatically construct the requirements-centric social networks and to apply the measures to them would motivate and facilitate the use of the framework by practitioners. The challenge of developing such tools is rooted on the vast diversity of software development tools used by software development practitioners. An automated solution would have to be able to deal with any type of data format and to integrate with diverse technological market demands in order to be portable to a wide number of companies. A return-of-investiment analysis of whether it is worth moving toward developing such package of tools has to be conducted before a decision is made. Therefore, the wish of offering practitioners an automated solution for the use of the framework is only an intended goal for now.

Appendix A

Questionnaire Sample

This appendix presents a sample of the questionnaire customized for a certain team member of the APP project. Alternatives in the demographic section were the only variance in relation to the SHIP project.

Figure A.1 presents the front page of the questionnaire where instructions were introduced to the respondents. Figure A.2 presents the demographic questions which characterize the team members' attributes. Figure A.3 shows the communication questions customized according to the members assigned to work with the certain team member that the questionnaire belonged to, and with the requirements that one was assigned to work on. Note that question 11 collects social network data in the roster format, where a list of people is provided for the respondent' selection. In contrast, question 12 collects social network data in the fixed choice format, where the respondent is asked to nominate additional people he communicated with. These additional people characterize the emergent members. Figure A.4 displays the current awareness questions also customized per team members and requirements that one was assigned to work on. Questions 18 and 19 follow the same roster and fixed choice formats presented in the communication section. Figure A.5 presents the list of requirements that the respondent was assigned to work on. This list is customized per respondent.

Questionnaire:
Identifying how project team members communicate and maintain awareness in practice

A. INTRODUCTORY NOTE

This questionnaire is designed to identify how project team members stay aware of events that occur within a project that are related to their work. This questionnaire application is part of a research conducted by SEGAL (Software Engineering Global interAction Lab), a research group of University of Victoria (UVic), British Columbia, Canada, coordinated by Dr. Daniela Damian. It is being developed in partnership with Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Rio Grande do Sul, Brazil, and ORG Technology Center (Brazil I/T), under the ORG/PUCRS Agreement.

All the information you provide will be kept confidential. Data collected from this questionnaire will be used for research purposes only. In particular we have no intention of evaluating your answers personally. Project team members were recruited by Project Managers or directly by the Research group based on their involvement with the "APP" project.

Please, read instructions very carefully and provide answer for all questions. On average, it should take 45 minutes to complete the questionnaire. We thank for your collaboration with this research!

B. VOCABULARY

Local team: team you work locally - at the same building, city, and country.
Multi-site context: when a group of members actively collaborate on a common software/systems project separated by distance, time zone and culture. I.e, the project has a distributed scenario.
Remote team: team you work remotely, i.e. team who is located in a different country.
Project team: refers to a group of members working on the same project, for a period of time only. It does not matter if the members are located at the same local or are remote.
Rework: to modify the previous version of a software unit (e.g. requirement specification, code) in response to changes or to improvements opportunities identified during the project life-cycle.

C. QUESTIONS

Instructions:
1. Answer all questions considering your involvement on "**APP**" project.
2. A list of requirements is provided in the *Appendix* (Section D) at the end of this questionnaire. Please, use this list as reference to answer questions asking about requirements.

Figure A.1: Sample of the questionnaire's front page

1 - Demographic data

1. What is your full name?

2. How many years of professional experience working in multi-site context environment do you have?
☐ Over 7 years ☐ 4 to 7 years ☐ 1 to 3 years ☐ Less than 1 year

3. What is your working relationship with ORG?
☐ Employee ☐ Contractor ☐ Trainee/Intern ☐ Other:

If you chose "Contractor", please identify your company.
☐ Contractor 1 ☐ Contractor 2 ☐ Contractor ☐ Other:

4. How many years are you working on the current company?
☐ Over 7 years ☐ 4 to 7 years ☐ 1 to 3 years ☐ Less than 1 year

5. Where are you located geographically? ☐ Brazil ☐ US ☐ Other:

6. What is your primary work location?
☐ ORG Office ☐ Contractor 1 Office ☐ Contractor 2 Office ☐ Contractor 3 Office

7. Which is your primary role in the "APP" project? If you play more than one, indicate the roles by numbers. Use "1" for primary role, "2" for secondary role, and so on).

( )	Configuration Management Coordinator	( )	Requirements Analyst
( )	Development Leader	( )	System Architect
( )	Developer	( )	Technical Leader
( )	Environment Coordinator	( )	Test Leader
( )	Product Manager	( )	Tester
( )	Project Manager	( )	Other:

8. When did you join the project? Since:
☐ Envisioning phase
☐ Planning phase
☐ Developing phase
☐ Stabilizing phase

Figure A.2: Sample of the questionnaire's demographic questions

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 ☐ R1 ☐ R2 ☐ R1 ☐ R2 ☐ R1 ☐ R2 ☐ 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 ☐ R1 ☐ R2 ☐ R1 ☐ R2 ☐ R1 ☐ R2 ☐ 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 ☐ R1 ☐ R2 ☐ R1 ☐ R2 ☐ R1 ☐ R2 ☐ 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: ☐ R1 ☐ R2 ☐ Others: ☐ R1 ☐ R2 ☐ Others: ☐ R1 ☐ R2 ☐ Others: ☐ 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: ☐ R1 ☐ R2 ☐ Others: ☐ R1 ☐ R2 ☐ Others: ☐ R1 ☐ R2 ☐ Others: ☐ R1 ☐ R2 ☐ Others:

Figure A.3: Sample of the questionnaire's customized communication questions

3 - Awareness

Instructions: *For questions 18 to 26, please do not hesitate in acknowledging you are not aware of or you don't know about something. Mark "N/A" only for those situations you really didn't interacted with the person. For questions we request you to provide a list of names, there are extra pages you can use to provide more information if you need more space than we made available.*

18. How aware are you of the current set of tasks that the following project team members are working on that is related to your work? Mark "N/A" if you believe you are not working on a specific requirement.

For example, if you are "aware" that "Team member 1" is working in the same tasks related to R1 that you are allocated to work on, select "Aware" option in "Awareness level" column for "R1". And if you are "not aware" that "Team member 1" is working in "R2", select "Not aware" in "R2" line for "Team member 1".

Name of project team member	Requirement	Awareness level
Team member 1	R1	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R2	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
Team member 2	R1	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R2	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
Team member 3	R1	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R2	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware

19. For the same additional people you listed on Question 12, please indicate how aware are you what the person is doing that is related to your work. Mark "N/A" if you believe you don't work with this person in a specific requirement.

Name	Requirement	Awareness level
	R1	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R2	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R1	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R2	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R1	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware
	R2	☐ N/A ☐ Very unaware ☐ Not aware ☐ Aware ☐ Very aware

Figure A.4: Sample of the questionnaire's customized awareness questions

D. APPENDIX

Requirements list from "APP" project.

This list is the subset of requirements that concerns your work.

ID	Requirement name
R1	Requirement 1
R2	Requirement 2

Figure A.5: Sample of the questionnaire's customized requirements list

Appendix B

Detailed Results for the Shipping System Project

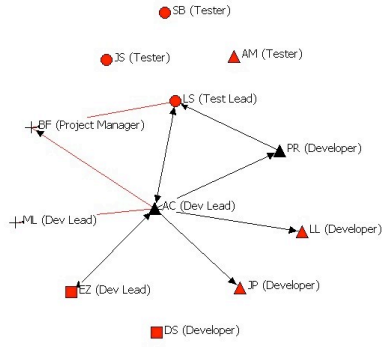
This appendix presents the data analysis results for the SHIP project in details. For each measure, the results obtained for the combination of each reason of communication and each requirements dependency set is presented.

B.1 (RQ2) RCSNs Characterization

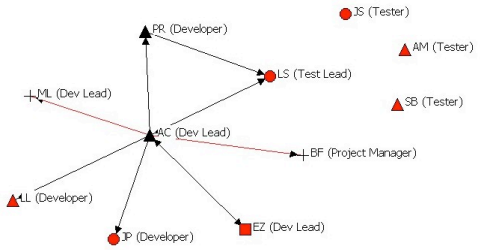
RCSN sociogram. The communication-RCSNs sociograms are organized by reason of communication. Figure B.1 displays the sociogram for the Requirements negotiation reason by requirements dependency set. Note that the sociogram for each reason of communication of the dependency set D_1 has been presented in Chapter 6, thus here D_2 to D_5 sets are presented. Figure B.2 presents the sociograms for the Requirements clarification reason, Figure B.3 for the Communication of changes reason, and Figure B.4 for the Coordination of activities reason.

The awareness-RCSNs sociograms for each set of requirements dependency is presented in Figure B.5. Similarly to the communication-RCSNs, the sociogram for the dependency set D_1 has been presented in Chapter 6.

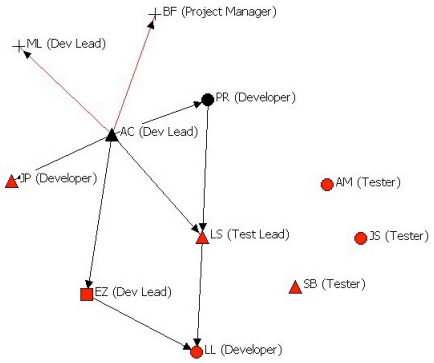
Ties statistics based on team members attributes. The communication-RCSNs ties statistics per reason of communication by set of requirements dependency is presented in Figure B.6, and it is organized as follows. The distribution of ties between assigned members is presented in Figure B.6(a), and between an assigned and an



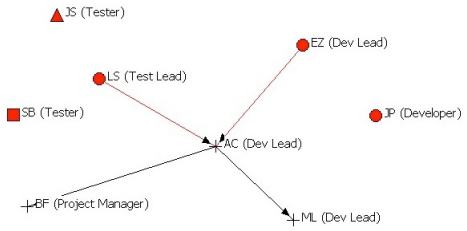
(a) Dependency set D_2



(b) Dependency set D_3



(c) Dependency set D_4



(d) Dependency set D_5

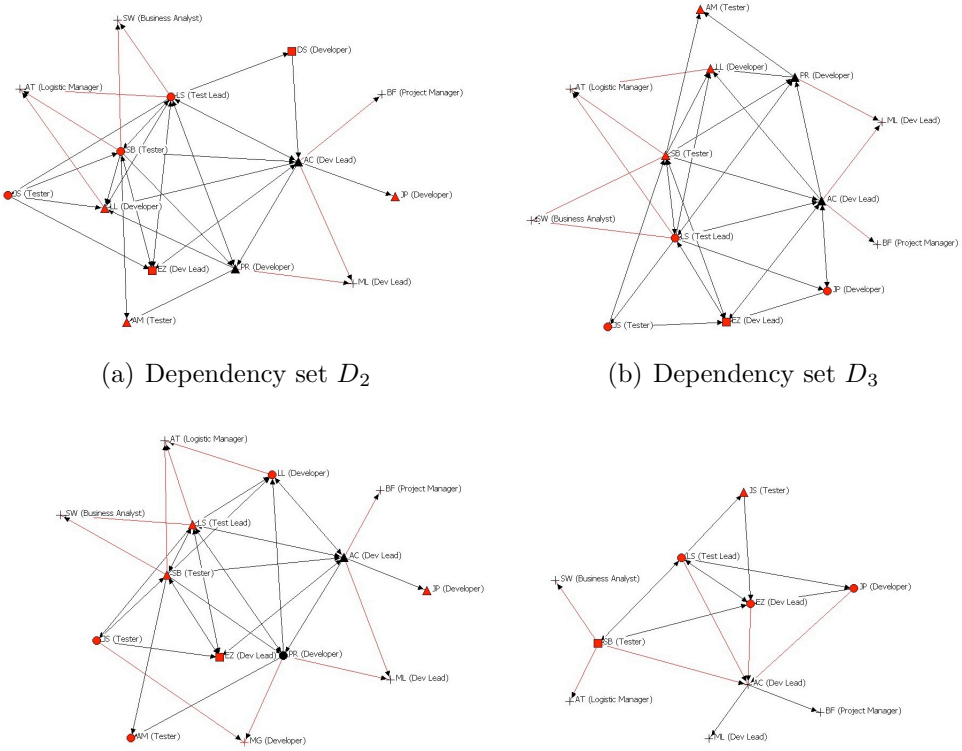
Figure B.1: SHIP's communication-RCSN sociograms for the Requirements Negotiation reason

emergent member in Figure B.6(b). The distribution of ties within-sites is presented in Figure B.6(c), and cross-sites in Figure B.6(d). The distribution of ties within-teams is presented in Figure B.6(e), and cross-teams in Figure B.6(f).

The awareness-RCSNs ties statistics by set of requirements dependency is presented in Figure B.7, and it is organized as follows. The distribution of ties between assigned members, and between an assigned and an emergent member is presented in Figure B.7(a). The distribution of ties by location is presented in Figure B.7(b), and by team in Figure B.7(c).

B.2 (RQ3) RCSNs Structures

Core-periphery. The list of team members belonging to the core and to the periphery of the communication-RCSNs for the Requirements negotiation reason is pre-



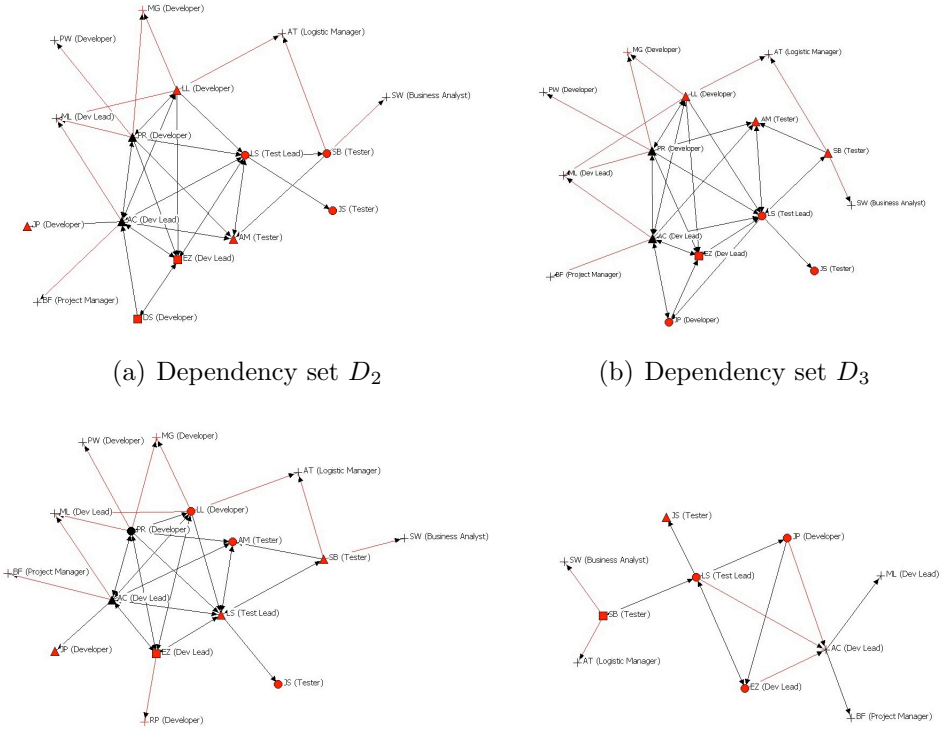
(a) Dependency set D_2 (b) Dependency set D_3

(c) Dependency set D_4 (d) Dependency set D_5

Figure B.2: SHIP's communication-RCSN sociograms for the Requirements Clarification reason

sented in Figure B.8. The list for the Requirements clarification reason is presented in Figure B.9. The list for the Communication of changes is in Figure B.10, and the list for the Coordination of activities reason is in Figure B.11. The list of team members belonging to the core and to the periphery of the awareness-RCSNs is presented in Figure B.12.

Clique. The list of team members by each identified clique for the Requirements negotiation reason is showed in Figure B.13. The list for the Requirements clarification list is showed in Figure B.14. The list for the Communication of changes reason is in Figure B.15, and the list for the Coordination of activities reason is in Figure B.16.



(a) Dependency set D_2

(b) Dependency set D_3

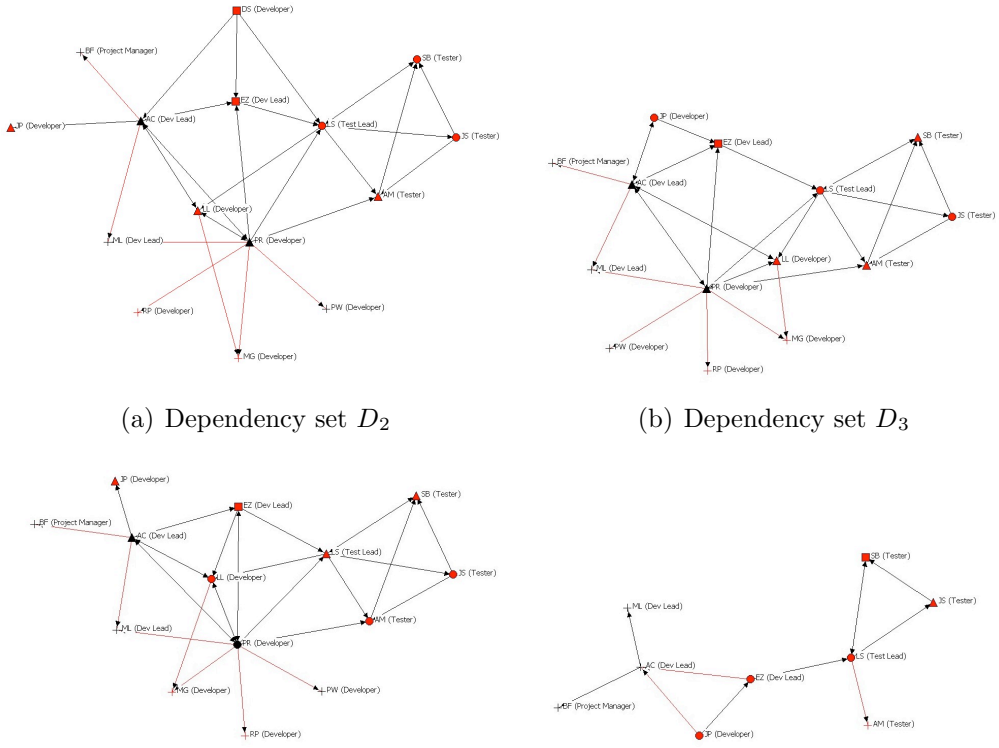
(c) Dependency set D_4

(d) Dependency set D_5

Figure B.3: SHIP's communication-RCSN sociograms for the Communication of Changes reason

B.3 (RQ4) RCSNs Information Flow

Component. The list of team members by each identified weak component is presented in Figure B.17. Note that the Requirements negotiation reason is the only topic of discussion with multiple components in each network. The list of team members by each identified strong component for the Requirements negotiation reasons is presented in Figure B.18, followed by the list for the Requirements clarification in Figure B.19. The list for the Communication of changes is in Figure B.20, and for the Coordination of activities reason is in Figure B.21.



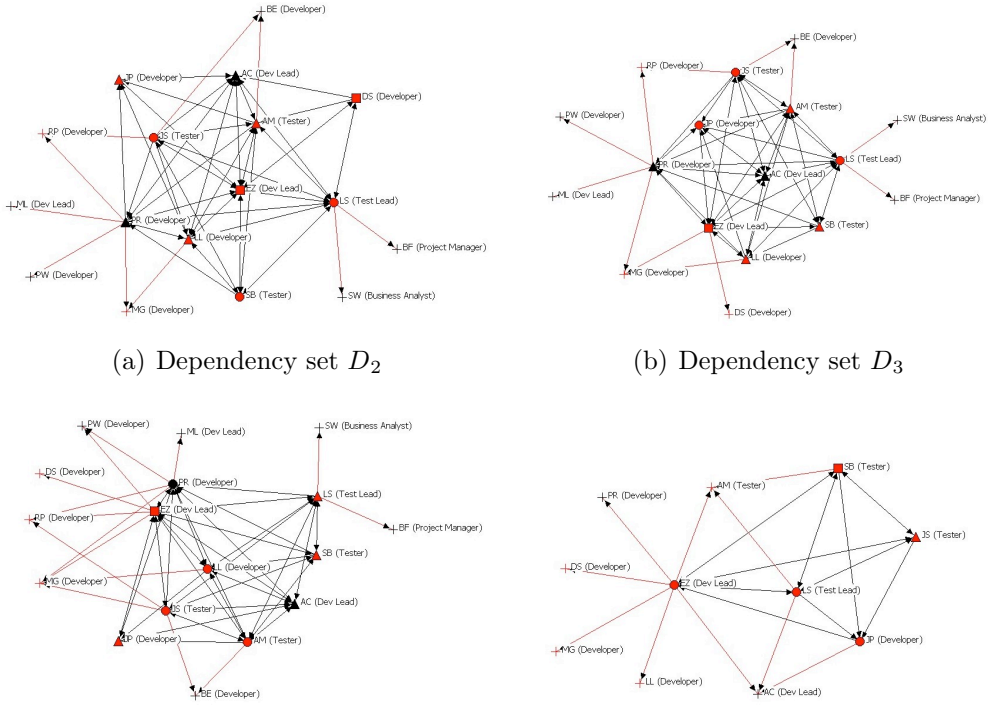
(a) Dependency set D_2

(b) Dependency set D_3

(c) Dependency set D_4

(d) Dependency set D_5

Figure B.4: SHIP's communication-RCSN sociograms for the Coordination of Activities reason



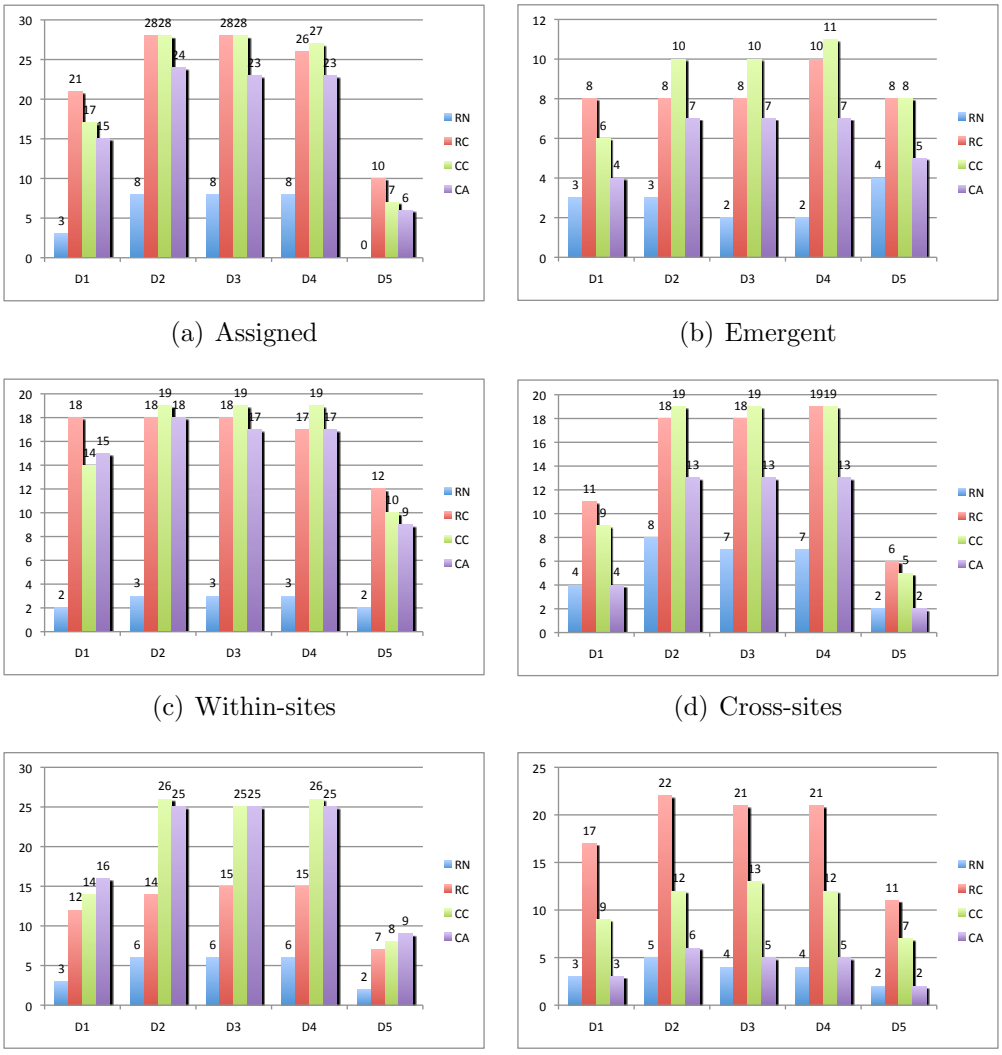
(a) Dependency set D_2

(b) Dependency set D_3

(c) Dependency set D_4

(d) Dependency set D_5

Figure B.5: SHIP's awareness-RCSN sociograms



(a) Assigned

(b) Emergent

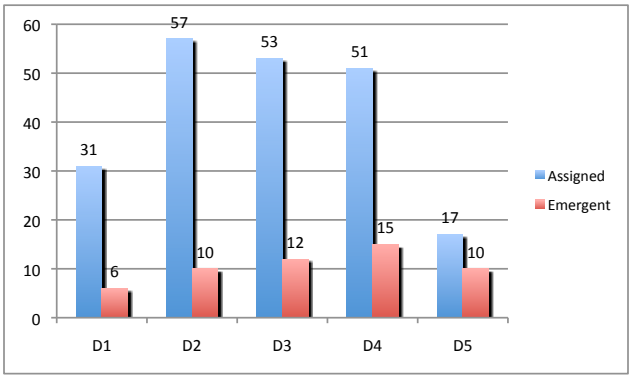
(c) Within-sites

(d) Cross-sites

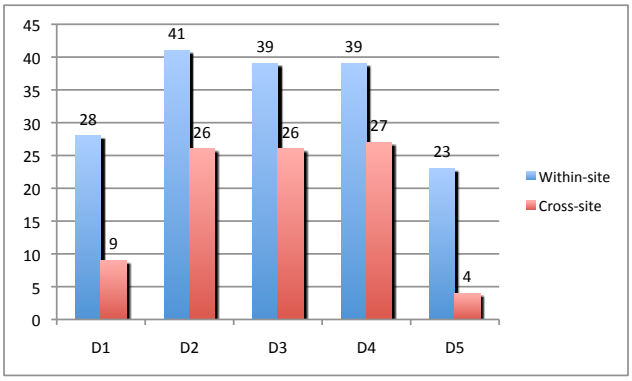
(e) Within-teams

(f) Cross-teams

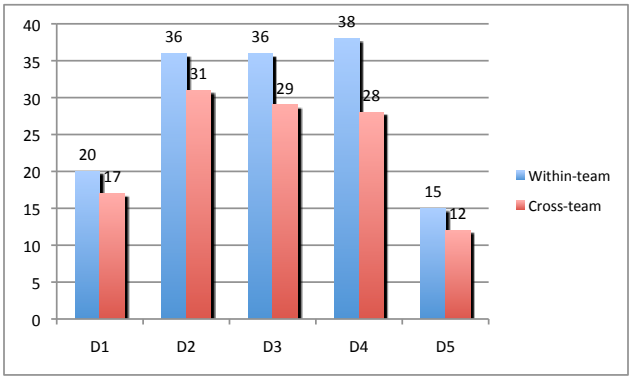
Figure B.6: SHIP's communication-RCSN ties statistics by requirements dependency set



(a) Assigned vs. Emergent



(b) Within- vs. Cross-sites



(c) Within- vs. Cross-teams

Figure B.7: SHIP's awareness-RCSN ties statistics by requirements dependency set

```
>> Requirements dependency set "D1":
Core:      AC (Dev Lead), EZ (Dev Lead)
Periphery: JS (Tester), AM (Tester), DS (Developer), LS (Test Lead),
           SB (Tester), BF (Project Manager), ML (Dev Lead)


>> Requirements dependency set "D2":
Core:      AC (Dev Lead), LL (Developer), EZ (Dev Lead),
           LS (Test Lead)
Periphery: JS (Tester), AM (Tester), DS (Developer), SB (Tester),
           JP (Developer), PR (Developer), BF (Project Manager),
           ML (Dev Lead)


>> Requirements dependency set "D3":
Core:      AC (Dev Lead), LL (Developer), EZ (Dev Lead),
           LS (Test Lead)
Periphery: JS (Tester), AM (Tester), SB (Tester), JP (Developer),
           PR (Developer), BF (Project Manager), ML (Dev Lead)


>> Requirements dependency set "D4":
Core:      AC (Dev Lead), EZ (Dev Lead), LS (Test Lead)
Periphery: JS (Tester), AM (Tester), LL (Developer), SB (Tester),
           JP (Developer), PR (Developer), BF (Project Manager),
           ML (Dev Lead)


>> Requirements dependency set "D5":
Core:      EZ (Dev Lead), LS (Test Lead), ML (Dev Lead), AC (Dev Lead)
Periphery: JS (Tester), SB (Tester), JP (Developer),
           BF (Project Manager)
```

Figure B.8: SHIP's communication-RCSN core-periphery members' list for the Requirements Negotiation reason

```
>> Requirements dependency set "D1":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), EZ (Dev Lead),
           DS (Developer), LS (Test Lead), SB (Tester)
Periphery: BF (Project Manager), ML (Dev Lead), LL (Developer),
           SW (Business Analyst), AT (Logistic Manager)


>> Requirements dependency set "D2":
Core:      JS (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), LS (Test Lead) SB (Tester), PR (Developer)
Periphery: AM (Tester), DS (Developer), JP (Developer),
           BF (Project Manager), ML (Dev Lead),
           AT (Logistic Manager), SW (Business Analyst)


>> Requirements dependency set "D3":
Core:      JS (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
           LS (Test Lead), SB (Tester), JP (Developer),
           PR (Developer)
Periphery: AM (Tester), BF (Project Manager), ML (Dev Lead),
           AT (Logistic Manager), SW (Business Analyst)


>> Requirements dependency set "D4":
Core:      JS (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
           LS (Test Lead), SB (Tester), PR (Developer)
Periphery: AM (Tester), JP (Developer), BF (Project Manager),
           ML (Dev Lead), MG (Developer), AT (Logistic Manager),
           SW (Business Analyst)


>> Requirements dependency set "D5":
Core:      JS (Tester), EZ (Dev Lead), SB (Tester), LS (Test Lead),
           JP (Developer), AC (Dev Lead)
Periphery: BF (Project Manager), ML (Dev Lead),
           AT (Logistic Manager), SW (Business Analyst)
```

Figure B.9: SHIP's communication-RCSN core-periphery members' list for the Requirements Clarification reason

```
>> Requirements dependency set "D1":
Core:      AM (Tester), AC (Dev Lead), EZ (Dev Lead), DS (Developer)
           LS (Test Lead)
Periphery: JS (Tester), SB (Tester), BF (Project Manager),
           ML (Dev Lead), LL (Developer), AT (Logistic Manager),
           SW (Business Analyst)


>> Requirements dependency set "D2":
Core:      AM (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
           LS (Test Lead), PR (Developer)
Periphery: JS (Tester), DS (Developer), SB (Tester), JP (Developer),
           BF (Project Manager), ML (Dev Lead),
           AT (Logistic Manager), MG (Developer), PW (Developer),
           SW (Business Analyst)


>> Requirements dependency set "D3":
Core:      AM (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), LS (Test Lead), PR (Developer)
Periphery: JS (Tester), SB (Tester), JP (Developer),
           BF (Project Manager), ML (Dev Lead),
           AT (Logistic Manager), MG (Developer), PW (Developer),
           SW (Business Analyst)


>> Requirements dependency set "D4":
Core:      AM (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
           LS (Test Lead), PR (Developer)
Periphery: JS (Tester), SB (Tester), JP (Developer),
           BF (Project Manager), ML (Dev Lead), RP (Developer),
           AT (Logistic Manager), MG (Developer), PW (Developer),
           SW (Business Analyst)


>> Requirements dependency set "D5":
Core:      JS (Tester), EZ (Dev Lead), SB (Tester), LS (Test Lead),
           JP (Developer), AC (Dev Lead)
Periphery: BF (Project Manager), ML (Dev Lead),
           AT (Logistic Manager), SW (Business Analyst)
```

Figure B.10: SHIP's communication-RCSN core-periphery members' list for the Communication of Changes reason

```
>> Requirements dependency set "D1":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), EZ (Dev Lead),
           DS (Developer), LS (Test Lead), SB (Tester)
Periphery: BF (Project Manager), ML (Dev Lead), LL (Developer)


>> Requirements dependency set "D2":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), DS (Developer), LS (Test Lead),
           SB (Tester), PR (Developer), ML (Dev Lead), MG (Developer)
Periphery: JP (Developer), BF (Project Manager), PW (Developer),
           RP (Developer)


>> Requirements dependency set "D3":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), LS (Test Lead), SB (Tester),
           JP (Developer), PR (Developer), ML (Dev Lead),
           MG (Developer)
Periphery: BF (Project Manager), PW (Developer), RP (Developer)


>> Requirements dependency set "D4":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), LS (Test Lead), SB (Tester),
           PR (Developer), ML (Dev Lead), MG (Developer)
Periphery: JP (Developer), BF (Project Manager), PW (Developer),
           RP (Developer)


>> Requirements dependency set "D5":
Core:      JS (Tester), EZ (Dev Lead), SB (Tester), LS (Test Lead)
Periphery: JP (Developer), AC (Dev Lead), BF (Project Manager),
           ML (Dev Lead), AM (Tester)
```

Figure B.11: SHIP's communication-RCSN core-periphery members' list for the Coordination of Activities reason

```
>> Requirements dependency set "D1":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), EZ (Dev Lead),
           DS (Developer), LS (Test Lead), SB (Tester)
Periphery: LL (Developer), PW (Developer), PR (Developer)


>> Requirements dependency set "D2":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), DS (Developer), LS (Test Lead), SB (Tester),
           PR (Developer)
Periphery: JP (Developer), BE (Developer), RP (Developer), BF (Project Manager),
           SW (Business Analyst), MG (Developer), ML (Dev Lead), PW (Developer)


>> Requirements dependency set "D3":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), LS (Test Lead), SB (Tester), JP (Developer),
           PR (Developer)
Periphery: BE (Developer), DS (Developer), MG (Developer), RP (Developer),
           BF (Project Manager), SW (Business Analyst), ML (Dev Lead),
           PW (Developer)


>> Requirements dependency set "D4":
Core:      JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
           EZ (Dev Lead), LS (Test Lead), SB (Tester), JP (Developer),
           PR (Developer)
Periphery: BE (Developer), DS (Developer), MG (Developer), PW (Developer),
           RP (Developer), BF (Project Manager), ML (Dev Lead),
           SW (Business Analyst)
```

Figure B.12: SHIP's awareness-RCSN core-periphery members' list

```
>> Requirements dependency set "D1":
Clique 1: AC (Dev Lead), LS (Test Lead), BF (Project Manager)


>> Requirements dependency set "D2":
Clique 1: AC (Dev Lead), LS (Test Lead), PR (Developer)
Clique 2: AC (Dev Lead), LS (Test Lead), BF (Project Manager)


>> Requirements dependency set "D3":
Clique 1: AC (Dev Lead), LS (Test Lead), PR (Developer)


>> Requirements dependency set "D4":
Clique 1: AC (Dev Lead), LS (Test Lead), PR (Developer)


>> Requirements dependency set "D5":
No clique found
```

Figure B.13: SHIP's communication-RCSN members' list by clique for the Requirements negotiation reason

```
>> Requirements dependency set "D1":
Clique 1: AC (Dev Lead), EZ (Dev Lead), DS (Developer), LS (Test Lead)
Clique 2: AM (Tester), AC (Dev Lead), EZ (Dev Lead), LS (Test Lead)
Clique 3: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), SB (Tester)
Clique 4: JS (Tester), EZ (Dev Lead), LS (Test Lead), SB (Tester)
Clique 5: LS (Test Lead), SB (Tester), SW (Business Analyst)

>> Requirements dependency set "D2":
Clique 1:  AC (Dev Lead), LL (Developer), LS (Test Lead), SB (Tester),
PR (Developer)
Clique 2:  AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), SB (Tester)
Clique 3:  AC (Dev Lead), DS (Developer), LS (Test Lead)
Clique 4:  AC (Dev Lead), PR (Developer), ML (Dev Lead)
Clique 5:  AM (Tester), SB (Tester), PR (Developer)
Clique 6:  JS (Tester), LL (Developer), LS (Test Lead), SB (Tester)
Clique 7:  JS (Tester), EZ (Dev Lead), LS (Test Lead), SB (Tester)
Clique 8:  LL (Developer), LS (Test Lead), SB (Tester),
           AT (Logistic Manager)
Clique 9:  LS (Test Lead), SB (Tester), SW (Business Analyst)

>> Requirements dependency set "D3":
Clique 1:  AC (Dev Lead), LL (Developer) ,LS (Test Lead), SB (Tester),
           PR (Developer)
Clique 2:  LL (Developer), LS (Test Lead), SB (Tester),
           AT (Logistic Manager)
Clique 3:  JS (Tester), EZ (Dev Lead), LS (Test Lead), SB (Tester)
Clique 4:  AC (Dev Lead), EZ (Dev Lead), LS (Test Lead) ,SB (Tester)
Clique 5:  LS (Test Lead), SB (Tester), SW (Business Analyst)
Clique 6:  AC (Dev Lead), EZ (Dev Lead), LS (Test Lead),
           JP (Developer)
Clique 7:  AM (Tester), SB (Tester), PR (Developer)
Clique 8:  AC (Dev Lead), PR (Developer), ML (Dev Lead)

>> Requirements dependency set "D4":
Clique 1:  AC (Dev Lead), LL (Developer), LS (Test Lead), SB (Tester),
           PR (Developer)
Clique 2:  AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), SB (Tester)
Clique 3:  JS (Tester), EZ (Dev Lead), LS (Test Lead), SB (Tester)
Clique 4:  LL (Developer), LS (Test Lead), SB (Tester),
           AT (Logistic Manager)
Clique 5:  LS (Test Lead), SB (Tester), SW (Business Analyst)
Clique 6:  AM (Tester), SB, (Tester), PR (Developer)
Clique 7:  AC (Dev Lead), PR (Developer), ML (Dev Lead)

>> Requirements dependency set "D5":
Clique 1:  EZ (Dev Lead), SB (Tester), LS (Test Lead), AC (Dev Lead)
Clique 2:  EZ (Dev Lead), LS (Test Lead), JP (Developer),
           AC (Dev Lead)
Clique 3:  JS (Tester), EZ (Dev Lead), LS (Test Lead)
```

Figure B.14: SHIP's communication-RCSN members' list by clique for the Requirements clarification reason

```
>> Requirements dependency set "D1":
Clique 1: AM (Tester), AC (Dev Lead), EZ (Dev Lead), LS (Test Lead)
Clique 2: AC (Dev Lead), EZ (Dev Lead), DS (Developer)
Clique 3: EZ (Dev Lead), DS (Developer), LL (Developer)


>> Requirements dependency set "D2":
Clique 1: AC (Dev Lead), LL (Developer), EZ (Dev Lead), LS (Test Lead),
          PR (Developer)
Clique 2: AC (Dev Lead), LL (Developer), PR (Developer), ML (Dev Lead)
Clique 3: AM (Tester), AC (Dev Lead), LS (Test Lead), PR (Developer)
Clique 4: AC (Dev Lead), EZ (Dev Lead), DS (Developer)
Clique 5: AM (Tester), LS (Test Lead), SB (Tester)
Clique 6: LL (Developer), PR (Developer), MG (Developer)


>> Requirements dependency set "D3":
Clique 1: AC (Dev Lead), LL (Developer), EZ (Dev Lead), LS (Test Lead),
          PR (Developer)
Clique 2: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), JP (Developer)
Clique 3: AM (Tester), AC (Dev Lead), LS (Test Lead), PR (Developer)
Clique 4: AC (Dev Lead), LL (Developer), PR (Developer), ML (Dev Lead)
Clique 5: AM (Tester), LS (Test Lead), SB (Tester)
Clique 6: LL (Developer), PR (Developer), MG (Developer)


>> Requirements dependency set "D4":
Clique 1: AC (Dev Lead), LL (Developer), EZ (Dev Lead), LS (Test Lead),
          PR (Developer)
Clique 2: AC (Dev Lead), LL (Developer), PR (Developer), ML (Dev Lead)
Clique 3: AM (Tester), AC (Dev Lead), LS (Test Lead), PR (Developer)
Clique 4: AM (Tester), LS (Test Lead), SB (Tester)
Clique 5: LL (Developer), PR (Developer), MG (Developer)


>> Requirements dependency set "D5":
Clique 1: EZ (Dev Lead), LS (Test Lead), JP (Developer), AC (Dev Lead)
```

Figure B.15: SHIP's communication-RCSN members' list by clique for the Communication of changes reason

```
>> Requirements dependency set "D1":
Clique 1:  JS (Tester), AM (Tester), LS (Test Lead)
Clique 2:  JS (Tester), LS (Test Lead), SB (Tester)
Clique 3:  EZ (Dev Lead), DS (Developer), LS (Test Lead)
Clique 4:  AC (Dev Lead), EZ (Dev Lead), DS (Developer)
Clique 5:  EZ (Dev Lead), DS (Developer), LL (Developer)


>> Requirements dependency set "D2":
Clique 1:  AC (Dev Lead), LL (Developer), PR (Developer)
Clique 2:  AC (Dev Lead), EZ (Dev Lead), PR (Developer)
Clique 3:  AC (Dev Lead), PR (Developer), ML (Dev Lead)
Clique 4:  AM (Tester), LS (Test Lead), PR (Developer)
Clique 5:  EZ (Dev Lead), LS (Test Lead), PR (Developer)
Clique 6:  LL (Developer), LS (Test Lead), PR (Developer)
Clique 7:  LL (Developer), PR (Developer), MG (Developer)
Clique 8:  AC (Dev Lead), EZ (Dev Lead), DS (Developer)
Clique 9:  EZ (Dev Lead), DS (Developer), LS (Test Lead)
Clique 10: JS (Tester), AM (Tester), LS (Test Lead), SB (Tester)


>> Requirements dependency set "D3":
Clique 1:  AC (Dev Lead), LL (Developer), PR (Developer)
Clique 2:  AC (Dev Lead), EZ (Dev Lead), PR (Developer)
Clique 3:  AC (Dev Lead), PR (Developer), ML (Dev Lead)
Clique 4:  AM (Tester), LS (Test Lead), PR (Developer)
Clique 5:  EZ (Dev Lead), LS (Test Lead), PR (Developer)
Clique 6:  LL (Developer), LS (Test Lead), PR (Developer)
Clique 7:  LL (Developer), PR (Developer), MG (Developer)
Clique 8:  JS (Tester), AM (Tester), LS (Test Lead), SB (Tester)
Clique 9:  AC (Dev Lead), EZ (Dev Lead), JP (Developer)


>> Requirements dependency set "D4":
Clique 1:  AC (Dev Lead), LL (Developer), EZ (Dev Lead),
           PR (Developer)
Clique 2:  LL (Developer), EZ (Dev Lead), LS (Test Lead),
           PR (Developer)
Clique 3:  LL (Developer), PR (Developer), MG (Developer)
Clique 4:  AM (Tester), LS (Test Lead), PR (Developer)
Clique 5:  AC (Dev Lead), PR (Developer), ML (Dev Lead)
Clique 6:  JS (Tester), AM (Tester), LS (Test Lead), SB (Tester)


>> Requirements dependency set "D5":
Clique 1:  JS (Tester), SB (Tester), LS (Test Lead)
Clique 2:  EZ (Dev Lead), JP (Developer), AC (Dev Lead)
```

Figure B.16: SHIP's communication-RCSN members' list by clique for the Coordination of activities reason

```
>> Requirements negotiation, requirements dependency set "D1":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: DS (Developer)
Component 4: SB (Tester)
Component 5: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead),
             BF (Project Manager), ML (Dev Lead)


>> Requirements negotiation, requirements dependency set "D2":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: DS (Developer)
Component 4: SB (Tester)
Component 5: AC (Dev Lead), LL (Developer), EZ (Dev Lead), LS (Test Lead),
             JP (Developer), PR (Developer), BF (Project Manager),
             ML (Dev Lead)


>> Requirements negotiation, requirements dependency set "D3":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: SB (Tester)
Component 4: AC (Dev Lead), LL (Developer), EZ (Dev Lead), LS (Test Lead),
             JP (Developer), PR (Developer), BF (Project Manager),
             ML (Dev Lead)


>> Requirements negotiation, requirements dependency set "D4":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: SB (Tester)
Component 4: AC (Dev Lead), LL (Developer), EZ (Dev Lead), LS (Test Lead),
             JP (Developer), PR (Developer), BF (Project Manager),
             ML (Dev Lead)


>> Requirements negotiation, requirements dependency set "D5":
Component 1: JS (Tester)
Component 2: SB (Tester)
Component 3: JP (Developer)
Component 4: EZ (Dev Lead), LS (Test Lead), BF (Project Manager),
             ML (Dev Lead), AC (Dev Lead)
```

Figure B.17: SHIP's communication-RCSN members' list by weak component

```
>> Requirements dependency set "D1":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: DS (Developer)
Component 4: LS (Test Lead)
Component 5: SB (Tester)
Component 6: BF (Project Manager)
Component 7: ML (Dev Lead)
Component 8: AC (Dev Lead), EZ (Dev Lead)

>> Requirements dependency set "D2":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: LL (Developer)
Component 4: DS (Developer)
Component 5: SB (Tester)
Component 6: JP (Developer)
Component 7: BF (Project Manager)
Component 8: ML (Dev Lead)
Component 9: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), PR (Developer)

>> Requirements dependency set "D3":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: LL (Developer)
Component 4: SB (Tester)
Component 5: JP (Developer)
Component 6: BF (Project Manager)
Component 7: ML (Dev Lead)
Component 8: AC (Dev Lead), EZ (Dev Lead), LS (Test Lead), PR (Developer)

>> Requirements dependency set "D4":
Component 1: JS (Tester)
Component 2: AM (Tester)
Component 3: AC (Dev Lead)
Component 4: LL (Developer)
Component 5: EZ (Dev Lead)
Component 6: LS (Test Lead)
Component 7: SB (Tester)
Component 8: JP (Developer)
Component 9: PR (Developer)
Component 10: BF (Project Manager)
Component 11: ML (Dev Lead)

>> Requirements dependency set "D5":
Component 1: JS (Tester)
Component 2: EZ (Dev Lead)
Component 3: SB (Tester)
Component 4: LS (Test Lead)
Component 5: JP (Developer)
Component 6: BF (Project Manager)
Component 7: ML (Dev Lead)
Component 8: AC (Dev Lead)
```

Figure B.18: SHIP's communication-RCSN members' list by strong component for the Requirements negotiation reason

```
>> Requirements dependency set "D1":
Component 1: BF (Project Manager)
Component 2: ML (Dev Lead)
Component 3: SW (Business Analyst)
Component 4: AT (Logistic Manager)
Component 5: JS (Tester), AM (Tester), AC (Dev Lead), EZ (Dev Lead),
             DS (Developer), LS (Test Lead), SB (Tester), LL (Developer)

>> Requirements dependency set "D2":
Component 1: AM (Tester)
Component 2: LL (Developer)
Component 3: JP (Developer)
Component 4: BF (Project Manager)
Component 5: ML (Dev Lead)
Component 6: AT (Logistic Manager)
Component 7: SW (Business Analyst)
Component 8: JS (Tester), AC (Dev Lead), EZ (Dev Lead), DS (Developer),
             LS (Test Lead), SB (Tester), PR (Developer)

>> Requirements dependency set "D3":
Component 1: AM (Tester)
Component 2: LL (Developer)
Component 3: BF (Project Manager)
Component 4: ML (Dev Lead)
Component 5: AT (Logistic Manager)
Component 6: SW (Business Analyst)
Component 7: JS (Tester), AC (Dev Lead), EZ (Dev Lead), LS (Test Lead),
             SB (Tester), JP (Developer), PR (Developer)

>> Requirements dependency set "D4":
Component 1: AM (Tester)
Component 2: JP (Developer)
Component 3: BF (Project Manager)
Component 4: ML (Dev Lead)
Component 5: MG (Developer)
Component 6: AT (Logistic Manager)
Component 7: SW (Business Analyst)
Component 8: JS (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
             LS (Test Lead), SB (Tester), PR (Developer)

>> Requirements dependency set "D5":
Component 1: AC (Dev Lead)
Component 2: BF (Project Manager)
Component 3: ML (Dev Lead)
Component 4: AT (Logistic Manager)
Component 5: SW (Business Analyst)
Component 6: JS (Tester), EZ (Dev Lead), SB (Tester), LS (Test Lead),
             JP (Developer)
```

Figure B.19: SHIP's communication-RCSN members' list by strong component for the Requirements clarification reason

```
>> Requirements dependency set "D1":
Component 1: JS (Tester)
Component 2: BF (Project Manager)
Component 3: ML (Dev Lead)
Component 4: AT (Logistic Manager)
Component 5: SW (Business Analyst)
Component 6: AM (Tester), AC (Dev Lead), EZ (Dev Lead), DS (Developer),
             LS (Test Lead), SB (Tester), LL (Developer)

>> Requirements dependency set "D2":
Component 1: JS (Tester)
Component 2: JP (Developer)
Component 3: BF (Project Manager)
Component 4: ML (Dev Lead)
Component 5: AT (Logistic Manager)
Component 6: MG (Developer)
Component 7: PW (Developer)
Component 8: SW (Business Analyst)
Component 9: AM (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
             DS (Developer), LS (Test Lead), SB (Tester), PR (Developer)

>> Requirements dependency set "D3":
Component 1: JS (Tester)
Component 2: BF (Project Manager)
Component 3: ML (Dev Lead)
Component 4: AT (Logistic Manager)
Component 5: MG (Developer)
Component 6: PW (Developer)
Component 7: SW (Business Analyst)
Component 8: AM (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
             LS (Test Lead), SB (Tester), JP (Developer), PR (Developer)

>> Requirements dependency set "D4":
Component 1: JS (Tester)
Component 2: JP (Developer)
Component 3: BF (Project Manager)
Component 4: ML (Dev Lead)
Component 5: RP (Developer)
Component 6: AT (Logistic Manager)
Component 7: MG (Developer)
Component 8: PW (Developer)
Component 9: SW (Business Analyst)
Component 10: AM (Tester), AC (Dev Lead), LL (Developer), EZ (Dev Lead),
              LS (Test Lead), SB (Tester), PR (Developer)

>> Requirements dependency set "D5":
Component 1: JS (Tester)
Component 2: AC (Dev Lead)
Component 3: BF (Project Manager)
Component 4: ML (Dev Lead)
Component 5: AT (Logistic Manager)
Component 6: SW (Business Analyst)
Component 7: EZ (Dev Lead), SB (Tester), LS (Test Lead), JP (Developer)
```

Figure B.20: SHIP's communication-RCSN members' list by strong component for the Communication of changes reason

Reachability. The communication-RCSNs reachability matrices for the Requirements negotiation reason per dependency set is showed in Figure B.22, followed by

```
>> Requirements dependency set "D1":
Component 1: BF (Project Manager)
Component 2: ML (Dev Lead)
Component 3: JS (Tester), AM (Tester), LS (Test Lead), SB (Tester)
Component 4: AC (Dev Lead), EZ (Dev Lead), DS (Developer), LL (Developer)

>> Requirements dependency set "D2":
Component 1: DS (Developer)
Component 2: JP (Developer)
Component 3: BF (Project Manager)
Component 4: ML (Dev Lead)
Component 5: MG (Developer)
Component 6: PW (Developer)
Component 7: RP (Developer)
Component 8: JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
             EZ (Dev Lead), LS (Test Lead), SB (Tester), PR (Developer)

>> Requirements dependency set "D3":
Component 1: BF (Project Manager)
Component 2: ML (Dev Lead)
Component 3: MG (Developer)
Component 4: PW (Developer)
Component 5: RP (Developer)
Component 6: JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
             EZ (Dev Lead), LS (Test Lead), SB (Tester), JP (Developer),
             PR (Developer)

>> Requirements dependency set "D4":
Component 1: JP (Developer)
Component 2: BF (Project Manager)
Component 3: ML (Dev Lead)
Component 4: MG (Developer)
Component 5: PW (Developer)
Component 6: RP (Developer)
Component 7: JS (Tester), AM (Tester), AC (Dev Lead), LL (Developer),
             EZ (Dev Lead), LS (Test Lead), SB (Tester), PR (Developer)

>> Requirements dependency set "D5":
Component 1: EZ (Dev Lead)
Component 2: JP (Developer)
Component 3: AC (Dev Lead)
Component 4: BF (Project Manager)
Component 5: ML (Dev Lead)
Component 6: AM (Tester)
Component 7: JS (Tester), SB (Tester), LS (Test Lead)
```

Figure B.21: SHIP's communication-RCSN members' list by strong component for the Coordination of activities reason

the Requirements clarification matrices in Figure B.23. The matrices for the Communication of changes reason is showed in Figure B.24 and for the Coordination of activities in Figure B.25.

Degree centrality. The communication-RCSNs outdegree and indegree centrality indexes by dependency set for the Requirements negotiation reason are presented

	J	A	A	E	D	L	S	B	M
JS (Tester)	0	0	0	0	0	0	0	0	0
AM (Tester)	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	0	0	0	1	0	1	0	1	1
EZ (Dev Lead)	0	0	1	0	0	1	0	1	1
DS (Developer)	0	0	0	0	0	0	0	0	0
LS (Test Lead)	0	0	0	0	0	0	0	1	0
SB (Tester)	0	0	0	0	0	0	0	0	0
BF (Project Manager)	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0

(a) Dependency set D_1

	J	A	A	L	E	D	L	S	J	P	B	M
JS (Tester)	0	0	0	0	0	0	0	0	0	0	0	0
AM (Tester)	0	0	0	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	0	0	0	1	1	0	1	0	1	1	1	1
LL (Developer)	0	0	0	0	0	0	0	0	0	0	0	0
EZ (Dev Lead)	0	0	1	1	0	0	1	0	1	1	1	1
DS (Developer)	0	0	0	0	0	0	0	0	0	0	0	0
LS (Test Lead)	0	0	1	1	1	0	0	0	1	1	1	1
SB (Tester)	0	0	0	0	0	0	0	0	0	0	0	0
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	0	0	1	1	1	0	1	0	1	0	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0

(b) Dependency set D_2

	J	A	A	L	E	L	S	J	P	B	M
JS (Tester)	0	0	0	0	0	0	0	0	0	0	0
AM (Tester)	0	0	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	0	0	0	1	1	1	0	1	1	1	1
LL (Developer)	0	0	0	0	0	0	0	0	0	0	0
EZ (Dev Lead)	0	0	1	1	0	1	0	1	1	1	1
LS (Test Lead)	0	0	1	1	1	0	0	1	1	1	1
SB (Tester)	0	0	0	0	0	0	0	0	0	0	0
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	0	0	1	1	1	1	0	1	0	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0

(c) Dependency set D_3

	J	A	A	L	E	L	S	J	P	B	M
JS (Tester)	0	0	0	0	0	0	0	0	0	0	0
AM (Tester)	0	0	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	0	0	1	1	1	1	0	1	1	1	1
LL (Developer)	0	0	0	0	0	0	0	0	0	0	0
EZ (Dev Lead)	0	0	0	1	0	0	0	0	0	0	0
LS (Test Lead)	0	0	0	1	0	0	0	0	0	0	0
SB (Tester)	0	0	0	0	0	0	0	0	0	0	0
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	0	0	0	1	0	1	0	0	0	0	0
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0

(d) Dependency set D_4

	J	E	S	L	J	B	M	A
JS (Tester)	0	0	0	0	0	0	0	0
EZ (Dev Lead)	0	0	0	0	0	1	1	1
SB (Tester)	0	0	0	0	0	0	0	0
LS (Test Lead)	0	0	0	0	0	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0
BF (Project Manager)	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0
AC (Dev Lead)	0	0	0	0	0	1	1	0

(e) Dependency set D_5

Figure B.22: SHIP's communication-RCSN reachability matrices for the Requirements Negotiation reason per dependency set

in Figure B.26. The degree indexes for the Requirements clarification reason are presented in Figure B.27. The indexes for the Communication of changes reason are showed in Figure B.28, and the indexes for the Coordination of activities reason is in Figure B.29. The awaress-RCSNs outdegree and indegree centrality indexes by dependency set are presented in Figure B.30.

	J	A	A	E	D	L	S	B	M	L	S	A
JS (Tester)	0	1	1	1	1	1	1	1	1	1	1	1
AM (Tester)	1	0	1	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	0	1	1	1	1	1	1	1	1
DS (Developer)	1	1	1	1	0	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0
LL (Developer)	1	1	1	1	1	1	1	1	1	0	1	1
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0

(a) Dependency set D_1

	J	A	A	L	E	D	L	S	J	P	B	M	A	S
JS (Tester)	0	1	1	1	1	1	1	1	1	1	1	1	1	1
AM (Tester)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	0	0	0	0	0	0	0	0	0	0	0	1	0	
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1
DS (Developer)	1	1	1	1	1	0	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	1	0	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	1	0	1	1	1	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	1	1	1	1	1	1	1	1	1	0	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Dependency set D_2

	J	A	A	L	E	L	S	J	P	B	M	A	S
JS (Tester)	0	1	1	1	1	1	1	1	1	1	1	1	1
AM (Tester)	0	0	0	0	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1
LL (Developer)	0	0	0	0	0	0	0	0	0	0	1	0	
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1	1
JP (Developer)	1	1	1	1	1	1	1	0	1	1	1	1	1
PR (Developer)	1	1	1	1	1	1	1	1	0	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0	0

(c) Dependency set D_3

	J	A	A	L	E	L	S	J	P	B	M	M	A	S
JS (Tester)	0	1	1	1	1	1	1	1	1	1	1	1	1	1
AM (Tester)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	1	1	1	1	1	1	1	1	0	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(d) Dependency set D_4

	J	A	A	L	E	L	S	J	P	B	M	M	A	S
JS (Tester)	0	1	1	1	1	1	1	1	1	1	1	1	1	1
AM (Tester)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	1	1	1	1	1	1	1	1	0	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(e) Dependency set D_5

Figure B.23: SHIP's communication-RCSN reachability matrices for the Requirements Clarification reason per dependency set

	J	A	A	E	D	L	S	B	M	L	A	S
JS (Tester)	0	0	0	0	0	0	0	0	0	0	0	0
AM (Tester)	1	0	1	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	0	1	1	1	1	1	1	1	1
DS (Developer)	1	1	1	1	0	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0
LL (Developer)	1	1	1	1	1	1	1	1	1	0	1	1
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0

(a) Dependency set D_1

	J	A	A	L	E	D	L	S	J	P	B	M	A	M	P	S
JS (Tester)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM (Tester)	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
DS (Developer)	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PW (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Dependency set D_2

	J	A	A	L	E	L	S	J	P	B	M	A	M	P	S
JS (Tester)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM (Tester)	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
JP (Developer)	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
PR (Developer)	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PW (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(c) Dependency set D_3

	J	A	A	L	E	L	S	J	P	B	M	R	A	M	P	S
JS (Tester)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM (Tester)	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PW (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(d) Dependency set D_4

	J	E	S	L	J	A	B	M	A	S
JS (Tester)	0	0	0	0	0	0	0	0	0	0
EZ (Dev Lead)	1	0	1	1	1	1	1	1	1	1
SB (Tester)	1	1	0	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	0	1	1	1	1	1	1
JP (Developer)	1	1	1	1	0	1	1	1	1	1
AC (Dev Lead)	0	0	0	0	0	0	1	1	0	0
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0
AT (Logistic Manager)	0	0	0	0	0	0	0	0	0	0
SW (Business Analyst)	0	0	0	0	0	0	0	0	0	0

(e) Dependency set D_5

Figure B.24: SHIP's communication-RCSN reachability matrices for the Communication of Changes reason per dependency set

	J	A	A	E	D	L	S	B	M	L
JS (Tester)	0	1	0	0	0	1	1	0	0	0
AM (Tester)	1	0	0	0	0	1	1	0	0	0
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	0	1	1	1	1	1	1
DS (Developer)	1	1	1	1	0	1	1	1	1	1
LS (Test Lead)	1	1	0	0	0	0	1	0	0	0
SB (Tester)	1	1	0	0	0	1	0	0	0	0
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0
LL (Developer)	1	1	1	1	1	1	1	1	1	0

(a) Dependency set D_1

	J	A	A	L	E	D	L	S	J	P	B	M	M	P	R
JS (Tester)	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1
AM (Tester)	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
DS (Developer)	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PW (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Dependency set D_2

	J	A	A	L	E	L	S	J	P	B	M	M	P	R
JS (Tester)	0	1	1	1	1	1	1	1	1	1	1	1	1	1
AM (Tester)	1	0	1	1	1	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1	1	1
JP (Developer)	1	1	1	1	1	1	1	0	1	1	1	1	1	1
PR (Developer)	1	1	1	1	1	1	1	1	0	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PW (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(c) Dependency set D_3

	J	A	A	L	E	L	S	J	P	B	M	M	P	R
JS (Tester)	0	1	1	1	1	1	1	1	1	1	1	1	1	1
AM (Tester)	1	0	1	1	1	1	1	1	1	1	1	1	1	1
AC (Dev Lead)	1	1	0	1	1	1	1	1	1	1	1	1	1	1
LL (Developer)	1	1	1	0	1	1	1	1	1	1	1	1	1	1
EZ (Dev Lead)	1	1	1	1	0	1	1	1	1	1	1	1	1	1
LS (Test Lead)	1	1	1	1	1	0	1	1	1	1	1	1	1	1
SB (Tester)	1	1	1	1	1	1	0	1	1	1	1	1	1	1
JP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PR (Developer)	1	1	1	1	1	1	1	1	0	1	1	1	1	1
BF (Project Manager)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MG (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PW (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP (Developer)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(d) Dependency set D_4

	J	E	S	L	J	A	B	M	A
JS (Tester)	0	0	1	1	0	0	0	0	1
EZ (Dev Lead)	1	0	1	1	0	1	1	1	1
SB (Tester)	1	0	0	1	0	0	0	0	1
LS (Test Lead)	1	0	1	0	0	0	0	0	1
JP (Developer)	1	1	1	1	0	1	1	1	1
AC (Dev Lead)	0	0	0	0	0	0	1	1	0
BF (Project Manager)	0	0	0	0	0	0	0	0	0
ML (Dev Lead)	0	0	0	0	0	0	0	0	0
AM (Tester)	0	0	0	0	0	0	0	0	0

(e) Dependency set D_5

Figure B.25: SHIP's communication-RCSN reachability matrices for the Coordination of Activities reason per dependency set

	OutDegree	InDegree
AC (Dev Lead)	4.000	1.000
LS (Test Lead)	1.000	1.000
EZ (Dev Lead)	1.000	1.000
JS (Tester)	0.000	0.000
DS (Developer)	0.000	0.000
AM (Tester)	0.000	0.000
SB (Tester)	0.000	0.000
BF (Project Manager)	0.000	2.000
ML (Dev Lead)	0.000	1.000

(a) Dependency set D_1

	OutDegree	InDegree
AC (Dev Lead)	7.000	2.000
LS (Test Lead)	2.000	2.000
EZ (Dev Lead)	1.000	1.000
PR (Developer)	1.000	1.000
AM (Tester)	0.000	0.000
DS (Developer)	0.000	0.000
JS (Tester)	0.000	0.000
SB (Tester)	0.000	0.000
JP (Developer)	0.000	1.000
LL (Developer)	0.000	1.000
BF (Project Manager)	0.000	2.000
ML (Dev Lead)	0.000	1.000

(b) Dependency set D_2

	OutDegree	InDegree
AC (Dev Lead)	7	2
PR (Developer)	1	1
LS (Test Lead)	1	2
EZ (Dev Lead)	1	1
AM (Tester)	0	0
JS (Tester)	0	0
SB (Tester)	0	0
JP (Developer)	0	1
LL (Developer)	0	1
BF (Project Manager)	0	1
ML (Dev Lead)	0	1

(c) Dependency set D_3

	OutDegree	InDegree
AC (Dev Lead)	6	0
PR (Developer)	1	1
LS (Test Lead)	1	2
EZ (Dev Lead)	1	1
AM (Tester)	0	0
JS (Tester)	0	0
SB (Tester)	0	0
JP (Developer)	0	1
LL (Developer)	0	2
BF (Project Manager)	0	1
ML (Dev Lead)	0	1

(d) Dependency set D_4

	OutDegree	InDegree
AC (Dev Lead)	2.000	2.000
EZ (Dev Lead)	1.000	0.000
LS (Test Lead)	1.000	0.000
JS (Tester)	0.000	0.000
SB (Tester)	0.000	0.000
JP (Developer)	0.000	0.000
ML (Dev Lead)	0.000	1.000
BF (Project Manager)	0.000	1.000

(e) Dependency set D_5

Figure B.26: SHIP's communication-RCSN degree centrality for the Requirements Negotiation reason per dependency set

	OutDegree	InDegree
AC (Dev Lead)	6.000	4.000
LS (Test Lead)	6.000	4.000
SB (Tester)	5.000	3.000
EZ (Dev Lead)	4.000	5.000
JS (Tester)	3.000	1.000
DS (Developer)	2.000	3.000
LL (Developer)	2.000	1.000
AM (Tester)	1.000	3.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	1.000
SW (Business Analyst)	0.000	2.000
AT (Logistic Manager)	0.000	1.000

(a) Dependency set D_1

	OutDegree	InDegree
LS (Test Lead)	9.000	4.000
SB (Tester)	8.000	3.000
AC (Dev Lead)	7.000	4.000
PR (Developer)	4.000	3.000
EZ (Dev Lead)	3.000	4.000
JS (Tester)	3.000	1.000
DS (Developer)	1.000	1.000
LL (Developer)	1.000	5.000
AM (Tester)	0.000	2.000
JP (Developer)	0.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	2.000
AT (Logistic Manager)	0.000	3.000
SW (Business Analyst)	0.000	2.000

(b) Dependency set D_2

	OutDegree	InDegree
LS (Test Lead)	9.000	4.000
SB (Tester)	8.000	3.000
AC (Dev Lead)	7.000	4.000
PR (Developer)	4.000	3.000
EZ (Dev Lead)	3.000	5.000
JS (Tester)	2.000	1.000
JP (Developer)	2.000	2.000
LL (Developer)	1.000	4.000
AM (Tester)	0.000	2.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	2.000
AT (Logistic Manager)	0.000	3.000
SW (Business Analyst)	0.000	2.000

(c) Dependency set D_3

	OutDegree	InDegree
SB (Tester)	8.000	3.000
LS (Test Lead)	8.000	4.000
AC (Dev Lead)	7.000	4.000
PR (Developer)	5.000	3.000
JS (Tester)	3.000	1.000
EZ (Dev Lead)	3.000	4.000
LL (Developer)	2.000	4.000
JP (Developer)	0.000	1.000
AM (Tester)	0.000	2.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	2.000
MG (Developer)	0.000	2.000
AT (Logistic Manager)	0.000	3.000
SW (Business Analyst)	0.000	2.000

(d) Dependency set D_4

	OutDegree	InDegree
SB (Tester)	5.000	2.000
LS (Test Lead)	5.000	2.000
EZ (Dev Lead)	3.000	4.000
AC (Dev Lead)	2.000	4.000
JP (Developer)	2.000	1.000
JS (Tester)	1.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	1.000
AT (Logistic Manager)	0.000	1.000
SW (Business Analyst)	0.000	1.000

(e) Dependency set D_5

Figure B.27: SHIP's communication-RCSN degree centrality for the Requirements Clarification reason per dependency set

	OutDegree	InDegree
AC (Dev Lead)	6.000	3.000
EZ (Dev Lead)	5.000	3.000
LS (Test Lead)	5.000	4.000
SB (Tester)	3.000	1.000
DS (Developer)	2.000	3.000
AM (Tester)	1.000	3.000
LL (Developer)	1.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	1.000
JS (Tester)	0.000	1.000
AT (Logistic Manager)	0.000	1.000
SW (Business Analyst)	0.000	1.000

(a) Dependency set D_1

	OutDegree	InDegree
AC (Dev Lead)	8.000	5.000
PR (Developer)	8.000	2.000
LL (Developer)	7.000	2.000
LS (Test Lead)	5.000	6.000
SB (Tester)	4.000	1.000
EZ (Dev Lead)	3.000	5.000
DS (Developer)	2.000	1.000
AM (Tester)	1.000	4.000
JS (Tester)	0.000	1.000
JP (Developer)	0.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	3.000
AT (Logistic Manager)	0.000	2.000
MG (Developer)	0.000	2.000
PW (Developer)	0.000	1.000
SW (Business Analyst)	0.000	1.000

(b) Dependency set D_2

	OutDegree	InDegree
PR (Developer)	8.000	2.000
AC (Dev Lead)	8.000	5.000
LL (Developer)	7.000	2.000
LS (Test Lead)	6.000	6.000
SB (Tester)	4.000	1.000
EZ (Dev Lead)	2.000	5.000
JP (Developer)	2.000	2.000
AM (Tester)	1.000	4.000
JS (Tester)	0.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	3.000
AT (Logistic Manager)	0.000	2.000
MG (Developer)	0.000	2.000
PW (Developer)	0.000	1.000
SW (Business Analyst)	0.000	1.000

(c) Dependency set D_3

	OutDegree	InDegree
PR (Developer)	8.000	3.000
AC (Dev Lead)	8.000	4.000
LL (Developer)	7.000	3.000
LS (Test Lead)	5.000	6.000
EZ (Dev Lead)	5.000	4.000
SB (Tester)	4.000	1.000
AM (Tester)	1.000	4.000
JP (Developer)	0.000	1.000
JS (Tester)	0.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	3.000
RP (Developer)	0.000	1.000
AT (Logistic Manager)	0.000	2.000
MG (Developer)	0.000	2.000
PW (Developer)	0.000	1.000
SW (Business Analyst)	0.000	1.000

(d) Dependency set D_4

	OutDegree	InDegree
LS (Test Lead)	5.000	2.000
SB (Tester)	3.000	1.000
AC (Dev Lead)	2.000	3.000
EZ (Dev Lead)	2.000	2.000
JP (Developer)	2.000	1.000
JS (Tester)	0.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	1.000
AT (Logistic Manager)	0.000	1.000
SW (Business Analyst)	0.000	1.000

(e) Dependency set D_5

Figure B.28: SHIP's communication-RCSN degree centrality for the Communication of Changes reason per dependency set

	OutDegree	InDegree
AC (Dev Lead)	4.000	2.000
EZ (Dev Lead)	3.000	2.000
LS (Test Lead)	3.000	4.000
DS (Developer)	3.000	2.000
AM (Tester)	2.000	2.000
JS (Tester)	2.000	2.000
SB (Tester)	1.000	2.000
LL (Developer)	1.000	1.000
ML (Dev Lead)	0.000	1.000
BF (Project Manager)	0.000	1.000

(a) Dependency set D_1

	OutDegree	InDegree
PR (Developer)	9.000	2.000
AC (Dev Lead)	6.000	4.000
LS (Test Lead)	4.000	4.000
DS (Developer)	3.000	0.000
LL (Developer)	3.000	3.000
EZ (Dev Lead)	2.000	3.000
JS (Tester)	2.000	1.000
AM (Tester)	1.000	3.000
SB (Tester)	1.000	3.000
JP (Developer)	0.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	2.000
MG (Developer)	0.000	2.000
PW (Developer)	0.000	1.000
RP (Developer)	0.000	1.000

(b) Dependency set D_2

	OutDegree	InDegree
PR (Developer)	9.000	2.000
AC (Dev Lead)	6.000	4.000
LS (Test Lead)	4.000	3.000
LL (Developer)	3.000	3.000
JS (Tester)	2.000	1.000
EZ (Dev Lead)	2.000	3.000
JP (Developer)	2.000	1.000
SB (Tester)	1.000	3.000
AM (Tester)	1.000	3.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	2.000
MG (Developer)	0.000	2.000
PW (Developer)	0.000	1.000
RP (Developer)	0.000	1.000

(c) Dependency set D_3

	OutDegree	InDegree
PR (Developer)	9.000	3.000
AC (Dev Lead)	6.000	3.000
EZ (Dev Lead)	4.000	2.000
LS (Test Lead)	4.000	3.000
LL (Developer)	3.000	4.000
JS (Tester)	2.000	1.000
SB (Tester)	1.000	3.000
AM (Tester)	1.000	3.000
JP (Developer)	0.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	2.000
MG (Developer)	0.000	2.000
PW (Developer)	0.000	1.000
RP (Developer)	0.000	1.000

(d) Dependency set D_4

	OutDegree	InDegree
LS (Test Lead)	3.000	2.000
JP (Developer)	2.000	0.000
EZ (Dev Lead)	2.000	1.000
AC (Dev Lead)	2.000	2.000
SB (Tester)	1.000	2.000
JS (Tester)	1.000	1.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	1.000
AM (Tester)	0.000	1.000

(e) Dependency set D_5

Figure B.29: SHIP's communication-RCSN degree centrality for the Coordination of Activities reason per dependency set

	OutDegree	InDegree
EZ (Dev Lead)	9.000	5.000
AM (Tester)	6.000	6.000
JS (Tester)	5.000	4.000
SB (Tester)	5.000	3.000
AC (Dev Lead)	4.000	3.000
LS (Test Lead)	4.000	6.000
DS (Developer)	4.000	4.000
LL (Developer)	0.000	4.000
PW (Developer)	0.000	1.000
PR (Developer)	0.000	1.000

(a) Dependency set D_1

	OutDegree	InDegree
JS (Tester)	9.000	4.000
AM (Tester)	9.000	7.000
LS (Test Lead)	9.000	7.000
PR (Developer)	8.000	6.000
EZ (Dev Lead)	7.000	8.000
LL (Developer)	7.000	7.000
AC (Dev Lead)	6.000	8.000
SB (Tester)	6.000	4.000
DS (Developer)	4.000	2.000
JP (Developer)	2.000	4.000
BE (Developer)	0.000	2.000
RP (Developer)	0.000	2.000
BF (Project Manager)	0.000	1.000
SW (Business Analyst)	0.000	1.000
MG (Developer)	0.000	2.000
ML (Dev Lead)	0.000	1.000
PW (Developer)	0.000	1.000

(b) Dependency set D_2

	OutDegree	InDegree
JS (Tester)	9.000	4.000
AM (Tester)	9.000	6.000
EZ (Dev Lead)	9.000	7.000
LS (Test Lead)	9.000	6.000
PR (Developer)	8.000	7.000
LL (Developer)	7.000	6.000
AC (Dev Lead)	6.000	7.000
SB (Tester)	6.000	4.000
JP (Developer)	2.000	6.000
BE (Developer)	0.000	2.000
DS (Developer)	0.000	1.000
MG (Developer)	0.000	3.000
RP (Developer)	0.000	2.000
BF (Project Manager)	0.000	1.000
SW (Business Analyst)	0.000	1.000
ML (Dev Lead)	0.000	1.000
PW (Developer)	0.000	1.000

(c) Dependency set D_3

	OutDegree	InDegree
EZ (Dev Lead)	11.000	7.000
AM (Tester)	9.000	6.000
JS (Tester)	9.000	4.000
LS (Test Lead)	8.000	6.000
PR (Developer)	8.000	7.000
LL (Developer)	7.000	6.000
AC (Dev Lead)	6.000	7.000
SB (Tester)	6.000	4.000
JP (Developer)	2.000	4.000
BE (Developer)	0.000	2.000
DS (Developer)	0.000	1.000
MG (Developer)	0.000	4.000
PW (Developer)	0.000	2.000
RP (Developer)	0.000	3.000
BF (Project Manager)	0.000	1.000
ML (Dev Lead)	0.000	1.000
SW (Business Analyst)	0.000	1.000

(d) Dependency set D_4

	OutDegree	InDegree
EZ (Dev Lead)	9	4
LS (Test Lead)	6	3
SB (Tester)	5	3
JS (Tester)	4	3
JP (Developer)	2	3
AC (Dev Lead)	0	3
AM (Tester)	0	3
DS (Developer)	0	1
LL (Developer)	0	1
MG (Developer)	0	1
PR (Developer)	0	1

(e) Dependency set D_5

Figure B.30: SHIP's awareness-RCSN degree centrality per dependency set

Appendix C

Detailed Results for the Support Applications Project

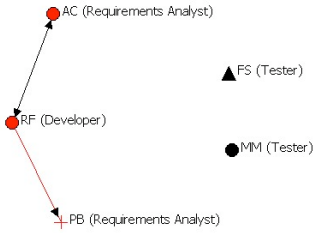
This appendix presents the data analysis results for the APP project in details. For each measure, the results obtained for the combination of each reason of communication and each requirements dependency set is presented.

C.1 (RQ2) RCSNs Characterization

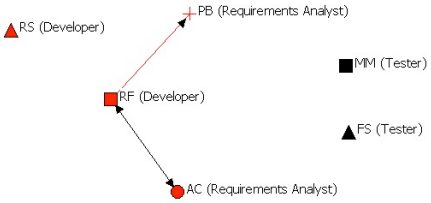
RCSN sociogram. The communication-RCSNs sociograms are organized by reason of communication. Figure C.1 displays the sociogram for the Requirements negotiation reason by requirements dependency set. Note that the sociogram for each reason of communication of the dependency set D_1 has been presented in Chapter 7, thus here D_2 to D_4 sets are presented. Figure C.2 presents the sociograms for the Requirements clarification reason, Figure C.3 for the Communication of changes reason, and Figure C.4 for the Coordination of activities reason.

The awareness-RCSNs sociograms for each set of requirements dependency is presented in Figure C.5. Similarly to the communication-RCSNs, the sociogram for the dependency set D_1 has been presented in Chapter 7.

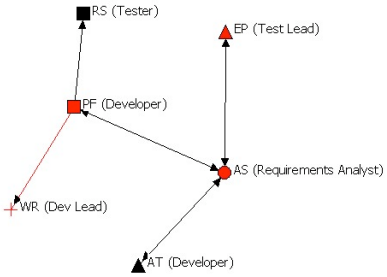
Ties statistics based on team members attributes. The communication-RCSNs ties statistics per reason of communication by set of requirements dependency is presented in Figure C.6, and it is organized as follows. The distribution of ties between assigned members is presented in Figure C.6(a), and between an assigned and an



(a) Dependency set D_2

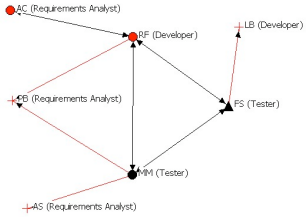


(b) Dependency set D_3

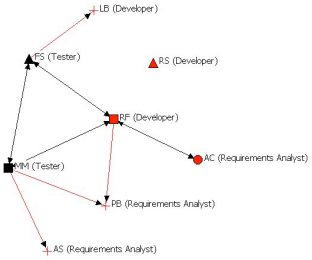


(c) Dependency set D_4

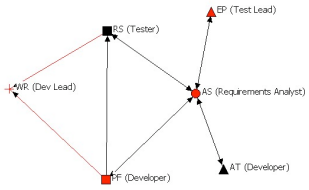
Figure C.1: APP's communication-RCSN sociograms for the Requirements Negotiation reason



(a) Dependency set D_2



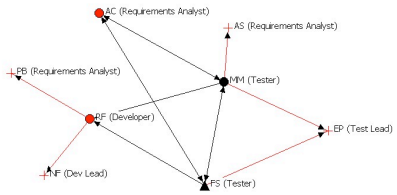
(b) Dependency set D_3



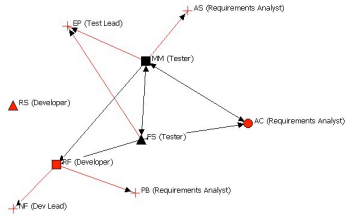
(c) Dependency set D_4

Figure C.2: APP's communication-RCSN sociograms for the Requirements Clarification reason

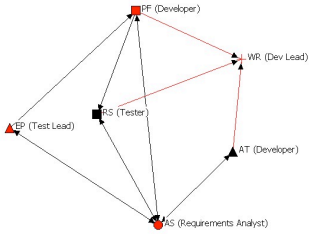
emergent member in Figure C.6(b). The distribution of ties within-offices is presented in Figure C.6(c), and cross-offices in Figure C.6(d). The distribution of ties within-teams is presented in Figure C.6(e), and cross-teams in Figure C.6(f).



(a) Dependency set D_2

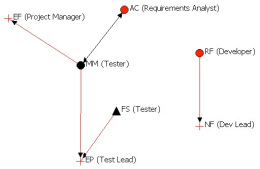


(b) Dependency set D_3

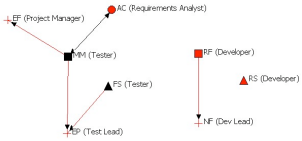


(c) Dependency set D_4

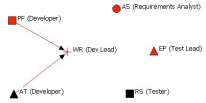
Figure C.3: APP's communication-RCSN sociograms for the Communication of Changes reason



(a) Dependency set D_2



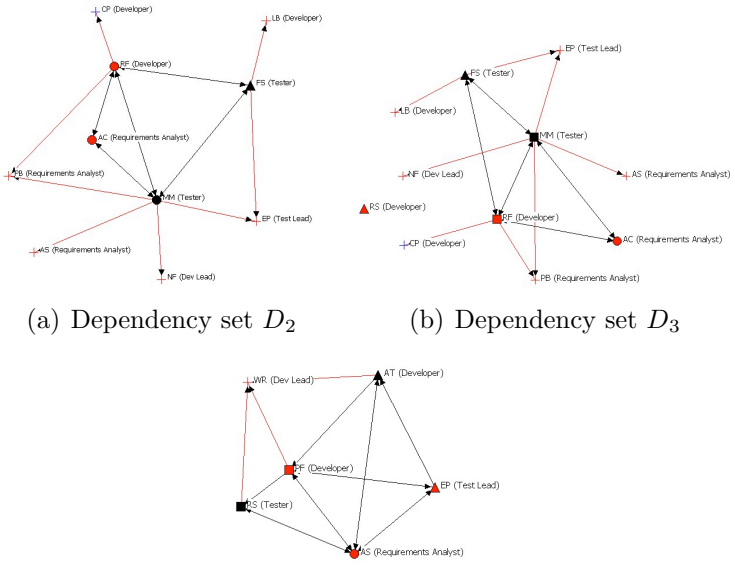
(b) Dependency set D_3



(c) Dependency set D_4

Figure C.4: APP's communication-RCSN sociograms for the Coordination of Activities reason

The awareness-RCSNs ties statistics by set of requirements dependency is presented in Figure C.7, and it is organized as follows. The distribution of ties between assigned members, and between an assigned and an emergent member is presented in Figure C.7(a). The distribution of ties by office location is presented in Figure C.7(b), and by team in Figure C.7(c).



(a) Dependency set D_2 (b) Dependency set D_3

(c) Dependency set D_4

Figure C.5: APP's awareness-RCSN sociograms

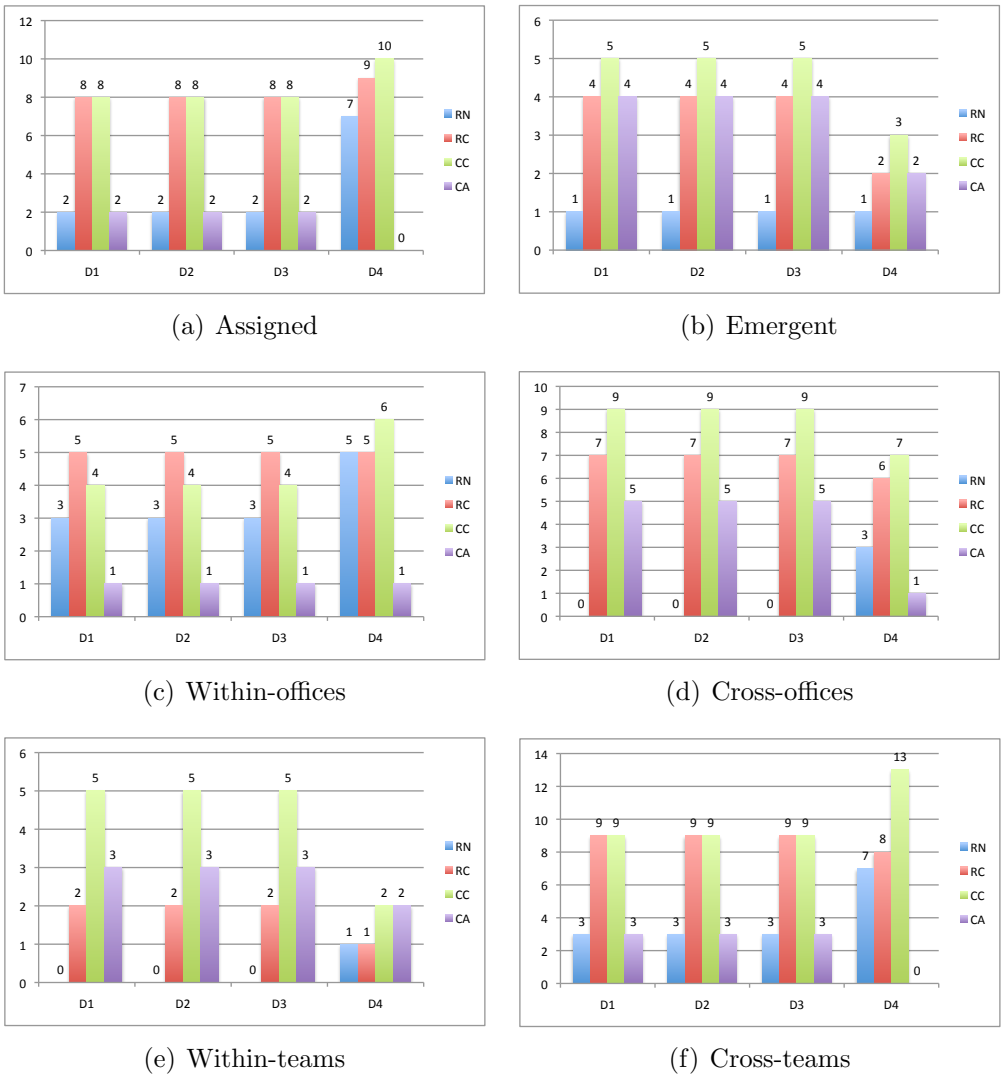
C.2 (RQ3) RCSNs Structures

Core-periphery. The list of team members belonging to the core and to the periphery of the communication-RCSNs for the Requirements negotiation reason is presented in Figure C.8. The list for the Requirements clarification reason is presented in Figure C.9. The list for the Communication of changes is in Figure C.10, and the list for the Coordination of activities reason is in Figure C.11. The list of team members belonging to the core and to the periphery of the awareness-RCSNs is presented in Figure C.12.

Clique. The list of team members by each identified clique for the Requirements clarification list is showed in Figure C.13, and the list for the Communication of changes reason is in Figure C.14. Note that no cliques were found for the Requirements negotiation and for the Coordination of activities reasons.

C.3 (RQ4) RCSNs Information Flow

Component. The list of team members by each identified weak component for the Requirements negotiation reasons is presented in Figure C.15, followed by the list for the Requirements clarification in Figure C.16. The list for the Communication of



(a) Assigned

(b) Emergent

(c) Within-offices

(d) Cross-offices

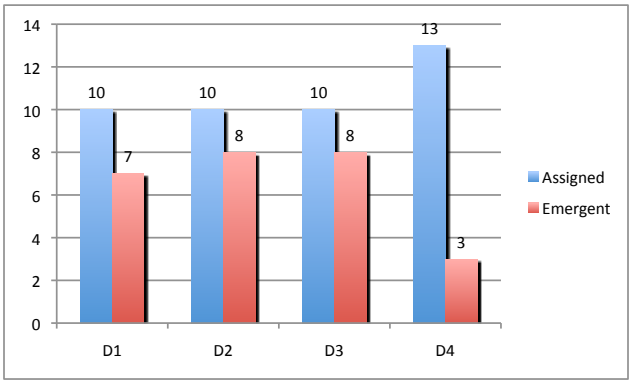
(e) Within-teams

(f) Cross-teams

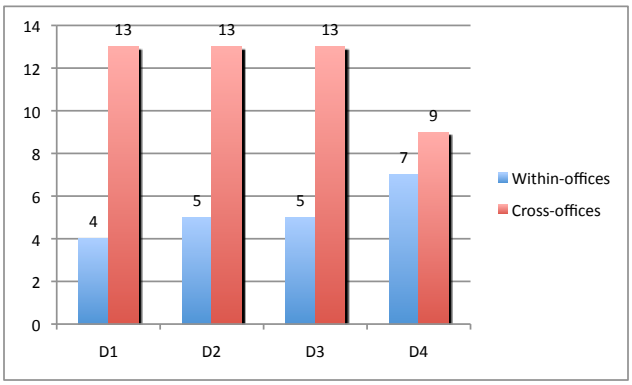
Figure C.6: APP's communication-RCSN ties statistics by requirements dependency set

changes is in Figure C.17, and for the Coordination of activities reason is in Figure C.18.

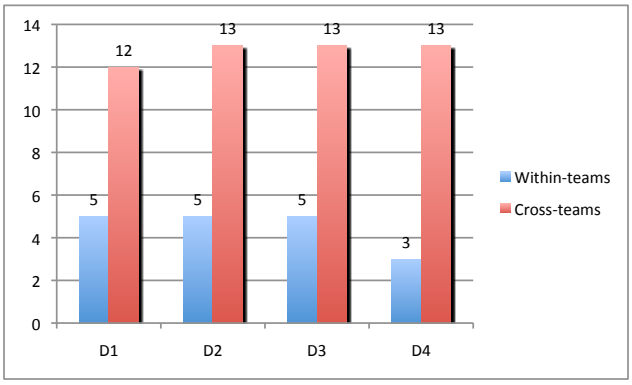
The list of team members by each identified strong component for the Requirements negotiation reasons is presented in Figure C.19, followed by the list for the Requirements clarification in Figure C.20. The list for the Communication of changes is in Figure C.21, and for the Coordination of activities reason is in Figure C.22.



(a) Assigned vs. Emergent



(b) Within- vs. Cross-offices



(c) Within- vs. Cross-teams

Figure C.7: APP's awareness-RCSN ties statistics by requirements dependency set

```
>> Requirements dependency set "D1":
Core:       RF (Developer), AC (Requirements Analyst)
Periphery:  FS (Tester), MM (Tester), RS (Developer),
            PB (Requirements Analyst)


>> Requirements dependency set "D2":
Core:      RF (Developer), AC (Requirements Analyst)
Periphery: FS (Tester), MM (Tester), PB (Requirements Analyst)


>> Requirements dependency set "D3":
Core:      RF (Developer), AC (Requirements Analyst)
Periphery: MM (Tester), FS (Tester), RS (Developer),
           PB (Requirements Analyst)


>> Requirements dependency set "D4":
Core:      AT (Developer), AS (Requirements Analyst), EP (Test Lead),
           PF (Developer)
Periphery: RS (Tester), WR (Dev Lead)
```

Figure C.8: APP's communication-RCSN core-periphery members' list for the Requirements Negotiation reason

```
>> Requirements dependency set "D1":
Core:      FS (Tester), RF (Developer), MM (Tester)
Periphery: AC (Requirements Analyst), RS (Developer), LB (Developer),
           AS (Requirements Analyst), PB (Requirements Analyst)


>> Requirements dependency set "D2":
Core:      FS (Tester), RF (Developer), MM (Tester)
Periphery: AC (Requirements Analyst), LB (Developer),
           AS (Requirements Analyst), PB (Requirements Analyst)


>> Requirements dependency set "D3":
Core:      FS (Tester), RF (Developer), MM (Tester)
Periphery: AC (Requirements Analyst), RS (Developer), LB (Developer),
           AS (Requirements Analyst), PB (Requirements Analyst)


>> Requirements dependency set "D4":
Core:      AT (Developer), AS (Requirements Analyst), EP (Test Lead),
           PF (Developer), RS (Tester)
Periphery: WR (Dev Lead)
```

Figure C.9: APP's communication-RCSN core-periphery members' list for the Requirements Clarification reason

```
>> Requirements dependency set "D1":
Core:      FS (Tester), RF (Developer), MM (Tester), AC (Requirements Analyst)
Periphery: RS (Developer), EP (Test Lead), AS (Requirements Analyst),
           NF (Dev Lead), PB (Requirements Analyst)


>> Requirements dependency set "D2":
Core:      FS (Tester), RF (Developer), MM (Tester), AC (Requirements Analyst)
Periphery: EP (Test Lead), AS (Requirements Analyst), NF (Dev Lead),
           PB (Requirements Analyst)


>> Requirements dependency set "D3":
Core:      FS (Tester), RF (Developer), MM (Tester), AC (Requirements Analyst)
Periphery: RS (Developer), EP (Test Lead), AS (Requirements Analyst),
           NF (Dev Lead), PB (Requirements Analyst)


>> Requirements dependency set "D4":
Core:      AT (Developer), AS (Requirements Analyst), EP (Test Lead),
           PF (Developer), RS (Tester)
Periphery: WR (Dev Lead)
```

Figure C.10: APP's communication-RCSN core-periphery members' list for the Communication of Changes reason

```
>> Requirements dependency set "D1":
Core:      MM (Tester), AC (Requirements Analyst)
Periphery: RS (Developer), FS (Tester), RF (Developer),
           EP (Test Lead), EF (Project Manager), NF (Dev Lead)


>> Requirements dependency set "D2":
Core:      MM (Tester), AC (Requirements Analyst)
Periphery: FS (Tester), RF (Developer), EP (Test Lead),
           EF (Project Manager), NF (Dev Lead)


>> Requirements dependency set "D3":
Core:      MM (Tester), AC (Requirements Analyst)
Periphery: RS (Developer), FS (Tester), RF (Developer),
           EP (Test Lead), EF (Project Manager), NF (Dev Lead)


>> Requirements dependency set "D4":
Core:      AT (Developer), WR (Dev Lead)
Periphery: AS (Requirements Analyst), EP (Test Lead),
           PF (Developer), RS (Tester)
```

Figure C.11: APP's communication-RCSN core-periphery members' list for the Coordination of Activities reason

```
>> Requirements dependency set "D1":
Core:      MM (Tester), AC (Requirements Analyst), FS (Tester), RF (Developer)
Periphery: RS (Developer), EP (Test Lead), LB (Developer), AS (Requirements Analyst),
           NF (Dev Lead), PB (Requirements Analyst), CP (Developer)


>> Requirements dependency set "D2":
Core:      FS (Tester), RF (Developer), MM (Tester), AC (Requirements Analyst)
Periphery: EP (Test Lead), LB (Developer), AS (Requirements Analyst), NF (Dev Lead),
           PB (Requirements Analyst), CP (Developer)


>> Requirements dependency set "D3":
Core:      FS (Tester), RF (Developer), MM (Tester), AC (Requirements Analyst)
Periphery: RS (Developer), EP (Test Lead), LB (Developer), AS (Requirements Analyst),
           NF (Dev Lead), PB (Requirements Analyst), CP (Developer)


>> Requirements dependency set "D4":
Core:      AT (Developer), AS (Requirements Analyst), EP (Test Lead), PF (Developer)
Periphery: RS (Tester), WR (Dev Lead)
```

Figure C.12: APP's awareness-RCSN core-periphery members' list

```
>> Requirements dependency set "D1":
Clique 1: FS (Tester), RF (Developer), MM (Tester)
Clique 2: RF (Developer), MM (Tester), PB (Requirements Analyst)


>> Requirements dependency set "D2":
Clique 1:  FS (Tester), RF (Developer), MM (Tester)
Clique 2:  RF (Developer), MM (Tester), PB (Requirements Analyst)


>> Requirements dependency set "D3":
Clique 1:  FS (Tester), RF (Developer), MM (Tester)
Clique 2:  RF (Developer), MM (Tester), PB (Requirements Analyst)


>> Requirements dependency set "D4":
Clique 1:  AS (Requirements Analyst), PF (Developer), RS (Tester)
Clique 2:  PF (Developer), RS (Tester), WR (Dev Lead)
```

Figure C.13: APP's communication-RCSN members' list by clique for the Requirements Clarification reason

```
>> Requirements dependency set "D1":
Clique 1: FS (Tester), RF (Developer), MM (Tester)
Clique 2: FS (Tester), MM (Tester), AC (Requirements Analyst)
Clique 3: FS (Tester), MM (Tester), EP (Test Lead)


>> Requirements dependency set "D2":
Clique 1: FS (Tester), RF (Developer), MM (Tester)
Clique 2: FS (Tester), MM (Tester), AC (Requirements Analyst)
Clique 3: FS (Tester), MM (Tester), EP (Test Lead)


>> Requirements dependency set "D3":
Clique 1: FS (Tester), RF (Developer), MM (Tester)
Clique 2: FS (Tester), MM (Tester), AC (Requirements Analyst)
Clique 3: FS (Tester), MM (Tester), EP (Test Lead)


>> Requirements dependency set "D4":
Clique 1: AS (Requirements Analyst), EP (Test Lead), PF (Developer)
Clique 2: AS (Requirements Analyst), PF (Developer), RS (Tester)
Clique 3: PF (Developer), RS (Tester), WR (Dev Lead)
```

Figure C.14: APP's communication-RCSN members' list by clique for the Communication of Changes reason

```
>> Requirements dependency set "D1":
Component 1: FS (Tester)
Component 2: MM (Tester)
Component 3: RS (Developer)
Component 4: RF (Developer), AC (Requirements Analyst),
             PB (Requirements Analyst)


>> Requirements dependency set "D2":
Component 1: FS (Tester)
Component 2: MM (Tester)
Component 3: RF (Developer), AC (Requirements Analyst),
             PB (Requirements Analyst)


>> Requirements dependency set "D3":
Component 1: MM (Tester)
Component 2: FS (Tester)
Component 3: RS (Developer)
Component 4: RF (Developer), AC (Requirements Analyst),
             PB (Requirements Analyst)


>> Requirements dependency set "D4":
Component 1: AT (Developer), AS (Requirements Analyst), EP (Test Lead),
             PF (Developer), RS (Tester), WR (Dev Lead)
```

Figure C.15: APP's communication-RCSN members' list by weak component for the Requirements Negotiation reason

```
>> Requirements dependency set "D1":
Component 1: RS (Developer)
Component 2: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst), LB (Developer),
             AS (Requirements Analyst), PB (Requirements Analyst)

>> Requirements dependency set "D2":
Component 1: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst), LB (Developer),
             AS (Requirements Analyst), PB (Requirements Analyst)


>> Requirements dependency set "D3":
Component 1: RS (Developer)
Component 2: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst), LB (Developer),
             AS (Requirements Analyst), PB (Requirements Analyst)


>> Requirements dependency set "D4":
Component 1: AT (Developer), AS (Requirements Analyst), EP (Test Lead),
             PF (Developer), RS (Tester), WR (Dev Lead)
```

Figure C.16: APP's communication-RCSN members' list by weak component for the Requirements Clarification reason

```
>> Requirements dependency set "D1":
Component 1: RS (Developer)
Component 2: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst), EP (Test Lead),
             AS (Requirements Analyst), NF (Dev Lead),
             PB (Requirements Analyst)

>> Requirements dependency set "D2":
Component 1: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst), EP (Test Lead),
             AS (Requirements Analyst), NF (Dev Lead),
             PB (Requirements Analyst)


>> Requirements dependency set "D3":
Component 1: RS (Developer)
Component 2: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst), EP (Test Lead),
             AS (Requirements Analyst), NF (Dev Lead),
             PB (Requirements Analyst)


>> Requirements dependency set "D4":
Component 1: AT (Developer), AS (Requirements Analyst), EP (Test Lead),
             PF (Developer), RS (Tester), WR (Dev Lead)
```

Figure C.17: APP's communication-RCSN members' list by weak component for the Communication of Changes reason

```
>> Requirements dependency set "D1":
Component 1: RS (Developer)
Component 2: RF (Developer)
Component 3: MM (Tester), AC (Requirements Analyst), FS (Tester),
             EP (Test Lead), EF (Project Manager)

>> Requirements dependency set "D2":
Component 1: RF (Developer), NF (Dev Lead)
Component 2: MM (Tester), AC (Requirements Analyst), FS (Tester),
             EP (Test Lead), EF (Project Manager)


>> Requirements dependency set "D3":
Component 1: RS (Developer)
Component 2: RF (Developer), NF (Dev Lead)
Component 3:  MM (Tester), AC (Requirements Analyst), FS (Tester),
              EP (Test Lead), EF (Project Manager)


>> Requirements dependency set "D4":
Component 1: AS (Requirements Analyst)
Component 2: EP (Test Lead)
Component 3: RS (Tester)
Component 4: AT (Developer), PF (Developer), WR (Dev Lead)
```

Figure C.18: APP's communication-RCSN members' list by weak component for the Coordination of Activities reason

```
>> Requirements dependency set "D1":
Component 1: RF (Developer), AC (Requirements Analyst)
Component 2: FS (Tester)
Component 3: MM (Tester)
Component 4: RS (Developer)
Component 5: PB (Requirements Analyst)


>> Requirements dependency set "D2":
Component 1: RF (Developer), AC (Requirements Analyst)
Component 2: FS (Tester)
Component 3: MM (Tester)
Component 4: PB (Requirements Analyst)


>> Requirements dependency set "D3":
Component 1: RF (Developer), AC (Requirements Analyst)
Component 2: MM (Tester)
Component 3: FS (Tester)
Component 4: RS (Developer)
Component 5: PB (Requirements Analyst)


>> Requirements dependency set "D4":
Component 1: RS (Tester)
Component 2: WR (Dev Lead)
Component 3: AT (Developer), AS (Requirements Analyst), EP (Test Lead),
             PF (Developer)
```

Figure C.19: APP's communication-RCSN members' list by strong component for the Requirements Negotiation reason

```
>> Requirements dependency set "D1":
Component 1: RS (Developer)
Component 2: LB (Developer)
Component 3: AS (Requirements Analyst)
Component 4: PB (Requirements Analyst)
Component 5: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst)


>> Requirements dependency set "D2":
Component 1: LB (Developer)
Component 2: AS (Requirements Analyst)
Component 3: PB (Requirements Analyst)
Component 4: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst)


>> Requirements dependency set "D3":
Component 1: RS (Developer)
Component 2: LB (Developer)
Component 3: AS (Requirements Analyst)
Component 4: PB (Requirements Analyst)
Component 5: FS (Tester), RF (Developer), MM (Tester),
             AC (Requirements Analyst)


>> Requirements dependency set "D4":
Component 1: WR (Dev Lead)
Component 2: AT (Developer), AS (Requirements Analyst), EP (Test Lead),
             PF (Developer), RS (Tester)
```

Figure C.20: APP's communication-RCSN members' list by strong component for the Requirements clarification reason

```
>> Requirements dependency set "D1":
Component 1: RF (Developer)
Component 2: RS (Developer)
Component 3: EP (Test Lead)
Component 4: AS (Requirements Analyst)
Component 5: NF (Dev Lead)
Component 6: PB (Requirements Analyst)
Component 7: FS (Tester), MM (Tester), AC (Requirements Analyst)


>> Requirements dependency set "D2":
Component 1: RF (Developer)
Component 2: EP (Test Lead)
Component 3: AS (Requirements Analyst)
Component 4: NF (Dev Lead)
Component 5: PB (Requirements Analyst)
Component 6: FS (Tester), MM (Tester), AC (Requirements Analyst)


>> Requirements dependency set "D3":
Component 1: RF (Developer)
Component 2: RS (Developer)
Component 3: EP (Test Lead)
Component 4: AS (Requirements Analyst)
Component 5: NF (Dev Lead)
Component 6: PB (Requirements Analyst)
Component 7: FS (Tester), MM (Tester), AC (Requirements Analyst)


>> Requirements dependency set "D4":
Component 1: WR (Dev Lead)
Component 2: AT (Developer), AS (Requirements Analyst), EP (Test Lead),
             PF (Developer), RS (Tester)
```

Figure C.21: APP's communication-RCSN members' list by strong component for the Communication of Changes reason

Reachability. The communication-RCSNs reachability matrices for the Requirements negotiation reason per dependency set is showed in Figure C.23, followed by the Requirements clarification matrices in Figure C.24. The matrices for the Communication of changes reason is showed in Figure C.25 and for the Coordination of activities in Figure C.26.

Degree centrality. The communication-RCSNs outdegree and indegree centrality indexes by dependency set for the Requirements negotiation reason are presented in Figure C.27. The degree indexes for the Requirements clarification reason are presented in Figure C.28. The indexes for the Communication of changes reason are showed in Figure C.29, and the indexes for the Coordination of activities reason is

```
>> Requirements dependency set "D1":
Component 1: MM (Tester), AC (Requirements Analyst)
Component 2: RS (Developer)
Component 3: FS (Tester)
Component 4: RF (Developer)
Component 5: EP (Test Lead)
Component 6: EF (Project Manager)
Component 7: NF (Dev Lead)


>> Requirements dependency set "D2":
Component 1: MM (Tester), AC (Requirements Analyst)
Component 2: FS (Tester)
Component 3: RF (Developer)
Component 4: EP (Test Lead)
Component 5: EF (Project Manager)
Component 6: NF (Dev Lead)


>> Requirements dependency set "D3":
Component 1: MM (Tester), AC (Requirements Analyst)
Component 2: RS (Developer)
Component 3: FS (Tester)
Component 4: RF (Developer)
Component 5: EP (Test Lead)
Component 6: EF (Project Manager)
Component 7: NF (Dev Lead)


>> Requirements dependency set "D4":
Component 1: AT (Developer)
Component 2: AS (Requirements Analyst)
Component 3: EP (Test Lead)
Component 4: PF (Developer)
Component 5: RS (Tester)
Component 6: WR (Dev Lead)
```

Figure C.22: APP's communication-RCSN members' list by strong component for the Coordination of Activities reason

in Figure C.30. The awaress-RCSNs outdegree and indegree centrality indexes by dependency set are presented in Figure C.31.

	R	A	F	M	R	P
RF (Developer)	0	1	0	0	0	1
AC (Requirements Analyst)	1	0	0	0	0	1
FS (Tester)	0	0	0	0	0	0
MM (Tester)	0	0	0	0	0	0
RS (Developer)	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0

(a) Dependency set D_1

	R	A	F	M	P
RF (Developer)	0	1	0	0	1
AC (Requirements Analyst)	1	0	0	0	1
FS (Tester)	0	0	0	0	0
MM (Tester)	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0

(b) Dependency set D_2

	R	A	M	F	R	P
RF (Developer)	0	1	0	0	0	1
AC (Requirements Analyst)	1	0	0	0	0	1
MM (Tester)	0	0	0	0	0	0
FS (Tester)	0	0	0	0	0	0
RS (Developer)	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0

(c) Dependency set D_3

	A	A	E	P	R	W
AT (Developer)	0	1	1	1	1	1
AS (Requirements Analyst)	1	0	1	1	1	1
EP (Test Lead)	1	1	0	1	1	1
PF (Developer)	1	1	1	0	1	1
RS (Tester)	0	0	0	0	0	0
WR (Dev Lead)	0	0	0	0	0	0

(d) Dependency set D_4

Figure C.23: APP's communication-RCSN reachability matrices for the Requirements Negotiation reason per dependency set

	F	R	M	A	L	A	P
FS (Tester)	0	1	1	1	1	1	1
RF (Developer)	1	0	1	1	1	1	1
MM (Tester)	1	1	0	1	1	1	1
AC (Requirements Analyst)	1	1	1	0	1	1	1
LB (Developer)	0	0	0	0	0	0	0
AS (Requirements Analyst)	0	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0	0

(a) Dependency set D_1

	F	R	M	A	L	A	P
FS (Tester)	0	1	1	1	1	1	1
RF (Developer)	1	0	1	1	1	1	1
MM (Tester)	1	1	0	1	1	1	1
AC (Requirements Analyst)	1	1	1	0	1	1	1
LB (Developer)	0	0	0	0	0	0	0
AS (Requirements Analyst)	0	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0	0

(b) Dependency set D_2

	F	R	M	A	R	L	A	P
FS (Tester)	0	1	1	1	0	1	1	1
RF (Developer)	1	0	1	1	0	1	1	1
MM (Tester)	1	1	0	1	0	1	1	1
AC (Requirements Analyst)	1	1	1	0	0	1	1	1
RS (Developer)	0	0	0	0	0	0	0	0
LB (Developer)	0	0	0	0	0	0	0	0
AS (Requirements Analyst)	0	0	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0	0	0

(c) Dependency set D_3

	A	A	E	P	R	W
AT (Developer)	0	1	1	1	1	1
AS (Requirements Analyst)	1	0	1	1	1	1
EP (Test Lead)	1	1	0	1	1	1
PF (Developer)	1	1	1	0	1	1
RS (Tester)	1	1	1	1	0	1
WR (Dev Lead)	0	0	0	0	0	0

(d) Dependency set D_4

Figure C.24: APP's communication-RCSN reachability matrices for the Requirements Clarification reason per dependency set

	F	R	M	A	R	E	A	N	P
FS (Tester)	0	1	1	1	0	1	1	1	1
RF (Developer)	0	0	0	0	0	0	0	1	1
MM (Tester)	1	1	0	1	0	1	1	1	1
AC (Requirements Analyst)	1	1	1	0	0	1	1	1	1
RS (Developer)	0	0	0	0	0	0	0	0	0
EP (Test Lead)	0	0	0	0	0	0	0	0	0
AS (Requirements Analyst)	0	0	0	0	0	0	0	0	0
NF (Dev Lead)	0	0	0	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0	0	0	0

(a) Dependency set D_1

	F	R	M	A	E	A	N	P
FS (Tester)	0	1	1	1	1	1	1	1
RF (Developer)	0	0	0	0	0	0	1	1
MM (Tester)	1	1	0	1	1	1	1	1
AC (Requirements Analyst)	1	1	1	0	1	1	1	1
EP (Test Lead)	0	0	0	0	0	0	0	0
AS (Requirements Analyst)	0	0	0	0	0	0	0	0
NF (Dev Lead)	0	0	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0	0	0

(b) Dependency set D_2

	F	R	M	A	R	E	A	N	P
FS (Tester)	0	1	1	1	0	1	1	1	1
RF (Developer)	0	0	0	0	0	0	0	1	1
MM (Tester)	1	1	0	1	0	1	1	1	1
AC (Requirements Analyst)	1	1	1	0	0	1	1	1	1
RS (Developer)	0	0	0	0	0	0	0	0	0
EP (Test Lead)	0	0	0	0	0	0	0	0	0
AS (Requirements Analyst)	0	0	0	0	0	0	0	0	0
NF (Dev Lead)	0	0	0	0	0	0	0	0	0
PB (Requirements Analyst)	0	0	0	0	0	0	0	0	0

(c) Dependency set D_3

	A	A	E	P	R	W
AT (Developer)	0	1	1	1	1	1
AS (Requirements Analyst)	1	0	1	1	1	1
EP (Test Lead)	1	1	0	1	1	1
PF (Developer)	1	1	1	0	1	1
RS (Tester)	1	1	1	1	0	1
WR (Dev Lead)	0	0	0	0	0	0

(d) Dependency set D_4

Figure C.25: APP's communication-RCSN reachability matrices for the Communication of Changes reason per dependency set

	M	A	R	F	R	E	E	N
MM (Tester)	0	1	0	0	0	1	1	0
AC (Requirements Analyst)	1	0	0	0	0	1	1	0
RS (Developer)	0	0	0	0	0	0	0	0
FS (Tester)	0	0	0	0	0	1	0	0
RF (Developer)	0	0	0	0	0	0	0	1
EP (Test Lead)	0	0	0	0	0	0	0	0
EF (Project Manager)	0	0	0	0	0	0	0	0
NF (Dev Lead)	0	0	0	0	0	0	0	0

(a) Dependency set D_1

	M	A	F	R	E	E	N
MM (Tester)	0	1	0	0	1	1	0
AC (Requirements Analyst)	1	0	0	0	1	1	0
FS (Tester)	0	0	0	0	1	0	0
RF (Developer)	0	0	0	0	0	0	1
EP (Test Lead)	0	0	0	0	0	0	0
EF (Project Manager)	0	0	0	0	0	0	0
NF (Dev Lead)	0	0	0	0	0	0	0

(b) Dependency set D_2

	M	A	R	F	R	E	E	N
MM (Tester)	0	1	0	0	0	1	1	0
AC (Requirements Analyst)	1	0	0	0	0	1	1	0
RS (Developer)	0	0	0	0	0	0	0	0
FS (Tester)	0	0	0	0	0	1	0	0
RF (Developer)	0	0	0	0	0	0	0	1
EP (Test Lead)	0	0	0	0	0	0	0	0
EF (Project Manager)	0	0	0	0	0	0	0	0
NF (Dev Lead)	0	0	0	0	0	0	0	0

(c) Dependency set D_3

	A	A	E	P	R	W
AT (Developer)	0	0	0	0	0	1
AS (Requirements Analyst)	0	0	0	0	0	0
EP (Test Lead)	0	0	0	0	0	0
PF (Developer)	0	0	0	0	0	1
RS (Tester)	0	0	0	0	0	0
WR (Dev Lead)	0	0	0	0	0	0

(d) Dependency set D_4

Figure C.26: APP's communication-RCSN reachability matrices for the Coordination of Activities reason per dependency set

	OutDegree	InDegree
RF (Developer)	2	1
AC (Requirements Analyst)	1	1
FS (Tester)	0	0
MM (Tester)	0	0
RS (Developer)	0	0
PB (Requirements Analyst)	0	1

(a) Dependency set D_1

	OutDegree	InDegree
RF (Developer)	2	1
AC (Requirements Analyst)	1	1
FS (Tester)	0	0
MM (Tester)	0	0
PB (Requirements Analyst)	0	1

(b) Dependency set D_2

	OutDegree	InDegree
RF (Developer)	2	1
AC (Requirements Analyst)	1	1
MM (Tester)	0	0
FS (Tester)	0	0
RS (Developer)	0	0
PB (Requirements Analyst)	0	1

(c) Dependency set D_3

	OutDegree	InDegree
PF (Developer)	3	1
AS (Requirements Analyst)	3	3
EP (Test Lead)	1	1
AT (Developer)	1	1
RS (Tester)	0	1
WR (Dev Lead)	0	1

(d) Dependency set D_4

Figure C.27: APP's communication-RCSN degree centrality for the Requirements Negotiation reason per dependency set

	OutDegree	InDegree
RF (Developer)	4.000	3.000
MM (Tester)	4.000	2.000
FS (Tester)	3.000	2.000
AC (Requirements Analyst)	1.000	1.000
LB (Developer)	0.000	1.000
AS (Requirements Analyst)	0.000	1.000
PB (Requirements Analyst)	0.000	2.000

(a) Dependency set D_1

	OutDegree	InDegree
RF (Developer)	4.000	3.000
MM (Tester)	4.000	2.000
FS (Tester)	3.000	2.000
AC (Requirements Analyst)	1.000	1.000
LB (Developer)	0.000	1.000
AS (Requirements Analyst)	0.000	1.000
PB (Requirements Analyst)	0.000	2.000

(b) Dependency set D_2

	OutDegree	InDegree
MM (Tester)	4.000	2.000
RF (Developer)	4.000	3.000
FS (Tester)	3.000	2.000
AC (Requirements Analyst)	1.000	1.000
RS (Developer)	0.000	0.000
LB (Developer)	0.000	1.000
AS (Requirements Analyst)	0.000	1.000
PB (Requirements Analyst)	0.000	2.000

(c) Dependency set D_3

	OutDegree	InDegree
AS (Requirements Analyst)	4	4
PF (Developer)	3	1
RS (Tester)	2	2
EP (Test Lead)	1	1
AT (Developer)	1	1
WR (Dev Lead)	0	2

(d) Dependency set D_4

Figure C.28: APP's communication-RCSN degree centrality for the Requirements Clarification reason per dependency set

	OutDegree	InDegree
MM (Tester)	5.000	2.000
FS (Tester)	4.000	2.000
RF (Developer)	2.000	2.000
AC (Requirements Analyst)	2.000	2.000
RS (Developer)	0.000	0.000
EP (Test Lead)	0.000	2.000
AS (Requirements Analyst)	0.000	1.000
NF (Dev Lead)	0.000	1.000
PB (Requirements Analyst)	0.000	1.000

(a) Dependency set D_1

	OutDegree	InDegree
MM (Tester)	5.000	2.000
FS (Tester)	4.000	2.000
RF (Developer)	2.000	2.000
AC (Requirements Analyst)	2.000	2.000
EP (Test Lead)	0.000	2.000
AS (Requirements Analyst)	0.000	1.000
NF (Dev Lead)	0.000	1.000
PB (Requirements Analyst)	0.000	1.000

(b) Dependency set D_2

	OutDegree	InDegree
MM (Tester)	5.000	2.000
FS (Tester)	4.000	2.000
RF (Developer)	2.000	2.000
AC (Requirements Analyst)	2.000	2.000
RS (Developer)	0.000	0.000
EP (Test Lead)	0.000	2.000
AS (Requirements Analyst)	0.000	1.000
NF (Dev Lead)	0.000	1.000
PB (Requirements Analyst)	0.000	1.000

(c) Dependency set D_3

	OutDegree	InDegree
AS (Requirements Analyst)	4	4
PF (Developer)	3	2
EP (Test Lead)	2	1
AT (Developer)	2	1
RS (Tester)	2	2
WR (Dev Lead)	0	3

(d) Dependency set D_4

Figure C.29: APP's communication-RCSN degree centrality for the Communication of Changes reason per dependency set

	OutDegree	InDegree
MM (Tester)	3.000	1.000
AC (Requirements Analyst)	1.000	1.000
RF (Developer)	1.000	0.000
FS (Tester)	1.000	0.000
RS (Developer)	0.000	0.000
EP (Test Lead)	0.000	2.000
EF (Project Manager)	0.000	1.000
NF (Dev Lead)	0.000	1.000

(a) Dependency set D_1

	OutDegree	InDegree
MM (Tester)	3.000	1.000
AC (Requirements Analyst)	1.000	1.000
FS (Tester)	1.000	0.000
RF (Developer)	1.000	0.000
EP (Test Lead)	0.000	2.000
EF (Project Manager)	0.000	1.000
NF (Dev Lead)	0.000	1.000

(b) Dependency set D_2

	OutDegree	InDegree
MM (Tester)	3.000	1.000
AC (Requirements Analyst)	1.000	1.000
RF (Developer)	1.000	0.000
FS (Tester)	1.000	0.000
RS (Developer)	0.000	0.000
EP (Test Lead)	0.000	2.000
EF (Project Manager)	0.000	1.000
NF (Dev Lead)	0.000	1.000

(c) Dependency set D_3

	OutDegree	InDegree
AT (Developer)	1	0
PF (Developer)	1	0
AS (Requirements Analyst)	0	0
EP (Test Lead)	0	0
RS (Tester)	0	0
WR (Dev Lead)	0	2

(d) Dependency set D_4

Figure C.30: APP's communication-RCSN degree centrality for the Coordination of Activities reason per dependency set

	OutDegree	InDegree
MM (Tester)	7	3
FS (Tester)	4	2
RF (Developer)	4	3
AC (Requirements Analyst)	2	2
RS (Developer)	0	0
EP (Test Lead)	0	2
LB (Developer)	0	1
AS (Requirements Analyst)	0	1
NF (Dev Lead)	0	1
PB (Requirements Analyst)	0	1
CP (Developer)	0	1

(a) Dependency set D_1

	OutDegree	InDegree
MM (Tester)	7.000	3.000
RF (Developer)	5.000	3.000
FS (Tester)	4.000	2.000
AC (Requirements Analyst)	2.000	2.000
EP (Test Lead)	0.000	2.000
LB (Developer)	0.000	1.000
AS (Requirements Analyst)	0.000	1.000
NF (Dev Lead)	0.000	1.000
PB (Requirements Analyst)	0.000	2.000
CP (Developer)	0.000	1.000

(b) Dependency set D_2

	OutDegree	InDegree
MM (Tester)	7	3
RF (Developer)	5	3
FS (Tester)	4	2
AC (Requirements Analyst)	2	2
RS (Developer)	0	0
EP (Test Lead)	0	2
LB (Developer)	0	1
AS (Requirements Analyst)	0	1
NF (Dev Lead)	0	1
PB (Requirements Analyst)	0	2
CP (Developer)	0	1

(c) Dependency set D_3

	OutDegree	InDegree
PF (Developer)	4	3
AS (Requirements Analyst)	4	4
EP (Test Lead)	3	2
AT (Developer)	3	2
RS (Tester)	2	2
WR (Dev Lead)	0	3

(d) Dependency set D_4

Figure C.31: APP's awareness-RCSN degree centrality per dependency set

Appendix D

The Complete Framework for Studying Requirements-Driven Collaboration

This appendix presents the complete and revisited version of the framework for studying requirements-driven collaboration aiming to allow its independent use for researchers or practitioners.

Requirements-driven collaboration is collaboration that occurs during the elicitation, definition, specification, implementation, testing, and management of requirements. To study requirements-driven collaboration, I propose a framework that uses concepts and measures from social network analysis [135] to obtain insights about the communication and awareness patterns of those involved in requirements-driven collaboration. The framework defines an investigative approach based on a social structure that focuses on the requirement as the unit of work around which collaboration occurs. I term this structure a requirements-centric team. The framework then consists of two parts:

- Part 1 defines the requirements-centric team and requirements-centric social network concepts, and
- Part 2 defines a number of social networks analysis measures to study aspects of requirements-driven collaboration.

D.1 Part 1. Defining the Concepts

D.1.1 Defining Requirements-Centric Teams

A **requirements-centric team** (RCT) is a cross-functional group whose members' work activities are related to one or more interrelated requirements, as well as downstream artifacts such as database design, source code and test cases. By *related to* I consider relationships such as *assigned to* and *communicating about*.

The membership of an RCT contains individuals that have a relationship to a requirement or multiple interrelated requirements. Such relationships also include relationships to downstream artifacts that trace to the requirement. Thus, the RCT membership includes individuals who work on project artifacts such as database design, source code and test cases, as well as individuals who send and receive communication artifacts such as e-mail and instant messages. As an example, consider a project team comprised of team members Bob, Eva, Frank, Geoff, Lisa, Ron and Todd, and a number of requirements R_1, R_2 and R_3. The following activities and relationships have been recorded: Lisa, a software designer, is writing a design specification implementing R_1, as well as test cases for R_1. Todd has written source code that implements R_1. Eva exchanged an e-mail message with Todd during their work about R_1. Consequently, the RCT associated with R_1 (R_1CT) contains Lisa, Todd, and Eva. This is illustrated in Figure D.1 that shows R_1 on the requirement plane, its associated project and communication artifacts on the artifact plane, and the R_1CT on the requirements-centric teams' plane.

A requirement-centric team can also provide a view of people who are working on multiple related requirements. If a requirement is related to another through requirement dependencies such as structural (e.g., refined-to, changes-to and similar-to dependencies), constraining (e.g., requires, and conflict-with dependencies) or cost/value (e.g., increases/decreases cost of dependencies) [38], the requirements-centric team associated to the interrelated requirements comprises all project members whose work activities relate to these requirements and their related downstream artifacts. Figure D.1 also illustrates the RCT associated to R_2 & R_3 (R_3 depends on R_2), so the $R_{2\&3}$ CT contains Eva, Todd, Ron, Bob, Geoff and Frank.

The RCT also applies to non-functional requirements. As non-functional requirements often have a relationship to functional requirements, and cross-cut many artifacts, an RCT can identify people who should collaborate because their work on

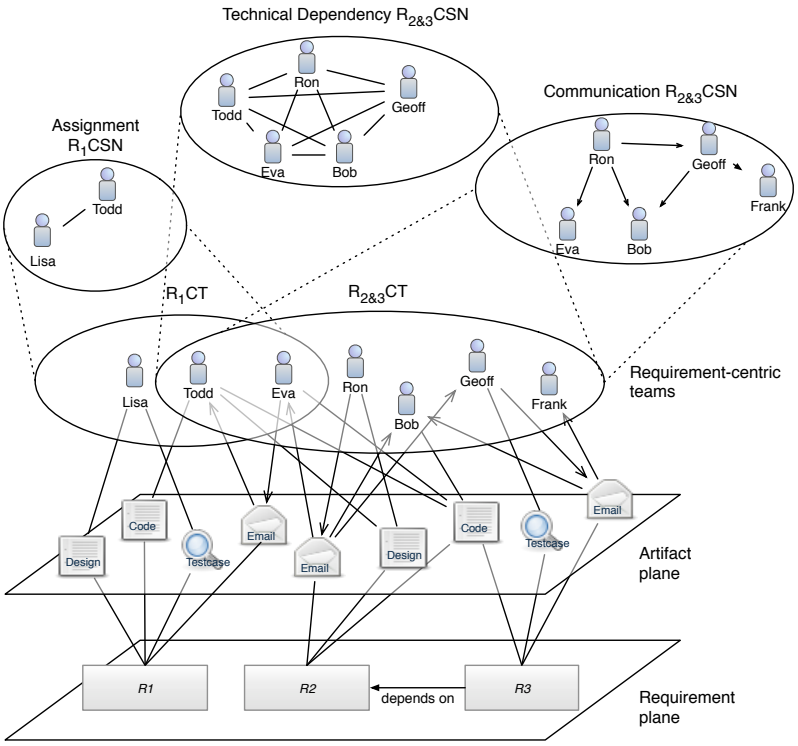


Figure D.1: Requirements-centric teams and different RCSNs

non-functional requirements influence those working on functional requirements.

D.1.2 Defining Requirements-Centric Social Networks

To analyze the collaboration within requirements-centric teams, I define a requirements-centric social network. A **requirements-centric social network** (RCSN) is a social network [135] that represents the members, also called *actors*, and relationships, also called *ties*, in a RCT. The actors in an RCSN are among the members of the RCT, and the ties in the network are representations of different relationships during these members' collaboration. For example, a tie can represent project members' requirements-related communication, assignment to work on the same requirements, contributions to the development of a requirement, or awareness of another's requirements-related work.

Representations such as social networks allow one to capture information about the real world relationships that form among people whose work is related to a requirement, and investigate questions such as *"Who has worked on artifacts related to particular requirements?"*, *"How does this compare to the project plan?"*, *"Who*

communicated or coordinated about these artifacts?", and *"Who are central people in the requirements-based communication and thus are key people in the processes of expertise seeking?"*.

Given specific research interests, one can define which types of relationships to represent in an RCSN, and collect appropriate data with to generate RCSNs containing different relationships. Examples of RCSNs that represent relationships include but are not limited to the following (Figure D.1 is used for illustration).

Technical dependency RCSN. A technical dependency RCSN contains members that should coordinate because there are technical dependencies among the artifacts they work on, e.g., those that contribute to the requirement and related downstream artifacts up to the current moment in time. This network is assumed to be fully connected, and ties are non-directional. In Figure D.1, there is a technical dependency between R_2 and R_3. Because Eva, Todd, Ron, and Bob are assigned to work on R_2, and Geoff on R_3, the technical dependency network contains all five team members. Such a network can be constructed using repository mining that identifies relationships between artifacts, such as call graphs and trace links.

This network can be useful for identifying how many project members have been involved in modifying the requirement or associated downstream artifacts. The information captured in this network can be used to propagate change information to members working on the requirement, and, more significantly, members working on dependent requirements. If one's work is affected by a dependent requirement, one has to receive information of changes about the related artifacts. Other uses for this network include expertise seeking to find members who recently worked on an artifact related to the requirement, and monitoring the amount of activity in the development, to identify requirements that may require additional resources.

Assignment RCSN. An assignment RCSN contains members from the RCT that have been assigned to work on the requirement or on its associated downstream artifacts. The network is assumed to be fully connected because it reflects technical dependencies among members who should coordinate with each other, and ties are non-directional. For example, in Figure D.1, Lisa is assigned to work on the design for R_1, and Todd is assigned to coding the modules related to R_1. Consequently, Todd and Lisa appear in the $assignment{-}R_1$CSN. Such a network can be constructed by extracting data from project planning or bug-tracking systems that contain informa-

tion about work assignment.

This network can be useful for identifying the expected scope of involvement and coordination in the development of a requirement. When constructed over a period of time, this network can show changes on allocation of members in a certain requirement and this information can be used by senior project management to restructure functional allocation of members in a department or in the company.

Communication RCSN. A communication RCSN contains members from the RCT that have communicated about the requirements or its associated downstream artifact. A directional tie is drawn if one person communicates about the requirement with another person. To construct a communication network, data can be automatically extracted from communication repositories such as mailing lists, online forum systems, instant messenger logs, and comments on bug-tracking systems, or self-reported by team members through daily log diaries and questionnaires.

This network can be useful for identifying communication activity generated around a requirement, and an indication of behaviors such as asking for clarifications on requirements and communication of changes. This network can be quite larger than the technical-dependency or assignment-RCSN in that it may include members who emerge as relevant to the coordination driven by the particular requirement – for instance, by having provided technical expertise – but who do not belong to the technical network because they have not modified any technical artifact or to the assignment network because they were not assigned to work on the related requirements or downstream artifacts. Frank is, for example, an emergent person in the $communication{-}R_{2\&3}CSN$ in Figure D.1 because Geoff is communicating with him and he was not assigned to work on R_2 or R_3. Similarly, fewer members than those in a technical-dependency or assignment relationship may be communicating during the project, indicating a possible lack of coordination in the development of the requirement. Figure D.1 shows Bob and Eva as not having communicated in a technical dependency relationship. Because there may be different reasons for communication, such as communication of changes [39], coordinating activities [107], and requesting clarification [108], one can construct and analyze networks that capture only a particular reason for communication. These reasons can be extracted from repositories or explicitly asked to members when data is collected through self-reported methods. It is important to be precise in the relationships mapped to clearly understand communication patterns and be able to improve collaboration [34].

Awareness RCSN. An awareness RCSN contains members from the RCT that have been identified to have awareness about other members and their work in the RCT. Awareness is the knowledge that one has about others and their working activities. Examples include knowledge of what is going on in a task in areas that affect that member's work [39] [136]; knowledge of which team members are around, where and when, as relevant for the task [52]; knowledge of how other members can help one in his work [48]; or knowledge of changes made on a project documentation artifact such as requirements specification. In the RCSN a directional tie is drawn if one person has awareness about the other, using the different types of awareness. To construct an awareness-based network, data can be collected through interviews or questionnaires. A question of the form *"Are you aware of this project member's current tasks?"* or *"Are you aware of how can these project members help you in your work on requirement R?"* can be asked of individuals in a project team. Data can also be collected through observation of team members in their work environment.

This network can be useful for identifying who in the organization is knowledgeable about activities that surround one's work. Since coordination of activities is a critical component of collaboration in requirements-centric teams, and awareness plays an important role in facilitating coordination, information about the extent to which members in an RCSN have awareness of each other's work is useful in diagnosing the collaboration ability of members in RCSNs. This network can be different than the communication network because people may become aware through other means than communication. For example, members developing code related to a requirement may stay aware of progress by subscribing to the code repository notification feature. On the other hand, a project manager may stay up-to-date about what is going on in the project by reading status report of member's activities.

D.2 Part 2. Defining Social Network Analysis Measures to Study Requirements-Driven Collaboration

Having defined requirements-centric social networks, a number of measures and tests from social network analysis was selected as mechanisms to explore the different aspects of requirements-driven collaboration. Table D.1 summarizes the questions

and insights that can be obtained by each measure adopted, and is organized by distinct aspects of requirements-driven collaboration that can be examined.

D.3 Analysis to Characterize Communication and Fleeting Knowledge in RDC

The measurements of RCSN network properties I propose can answer questions such as "What types of requirements require communication-based coordination?", "Which requirements required high amount of clarifications and may be then problematic because of unclear description?", and "Which requirements required high amount of coordination potentially because of changes to them?". In addition, characteristics of the members, such as role in an organization, level of experience and geographical location, may influence relationships observed in a RCSN. These characteristics are called *actor attributes*. By analyzing the attributes of RCSN members one can partition the network into smaller and more specific groups. By studying how information flows within and across groups one can study, for example, how frequently project RCT members communicate with those outside their team, or how frequently they communicate across distance.

Table D.1: Summary of measures by requirements-driven collaboration aspect

Characterization of Communication and Fleeting Knowledge in RDC		
Measure	*Question*	*Empirically-informed insight*
RCSN visualization	How does the RCSN look like?	Overall RDC insights
RCSN size	How many people are collaborating?	Number of members
RCSN density	How tightly-coupled is the RCT?	Cohesion
Ties statistics	What kind of interactions are present?	Patterns of collaboration
Communication and Fleeting Knowledge Network Structures in RDC		
Measure	*Question*	*Empirically-informed insight*
Centralization	Is collaboration centralized around certain members?	Knowledge distribution
Core-periphery	Who is at the core of the RCSN?	Key members
Ties reciprocity	Do members collaborate in a equal manner with each other?	Reciprocal collaboration
Clique	Who are the well-connected groups within the RCSN?	Key members
Information Flow Patterns in RDC		
Measure	*Question*	*Empirically-informed insight*
Component	Are all members of the RCT connected?	Extent of network division
Reachability	To what extent information can be shared with everyone?	Ability to reach all members
Cutpoints	Are there members that if removed would disrupt info flow?	Crucial members
Degree	Who are the central members?	Most active members
Brokerage	Who are the mediators of information?	Communication brokers
Alignment among Coord. Needs, Actual Coord., and Org. Structure in RDC		
Measure	*Question*	*Empirically-informed insight*
Alignment	To what extent are social relationships aligned reqs dep? To what extent RCTs follow the organization structure?	Ability to coordinate aligned with requirements dependencies Coordination behavior aligned organization structure

One can view the requirements-centric social network as consisting of different functional groups located at different geographical locations, or as groups of experts and novices. The actor attributes thus provides a useful dimension of analysis of "distance" between members in a RCSN. The geographical distance is an obvious one, but here I present other types of distance such as functional distance or level of experience. If two members are close in one dimension, they may consider themselves quite distant in other dimensions and make decisions about information sharing behavior. For example, a developer may exchange communication more frequently with geographically-distant senior developers than with novice developers in the same office. Thus, one can study relationships between attributes such as distance and functional role on patterns of information flow in RCSNs.

Visual composition of the entire RCSN and its members' attributes. A RCSN sociogram is a visual representation of the requirements-driven collaboration relationships established by members of a certain requirements-centric team. In addition to the RCT members (nodes) and relationships (ties) between them, a RCSN sociogram can also display its members attributes. The visualization of a sociogram that shows attributes can reveal patterns of relationship behavior among requirements-centric team members, bring hidden structural features to light, and help on the understanding of the communication and fleeting knowledge patterns that generated the network structure. For example, the node shape can represent the *member's role*, the actor color can indicate the *member's location*, and the actor size can indicate the *time that the member is working in a project*. The examination of a communication RCSN in terms of the three attributes allows one to partition the RCT into those who communicate within their teammates versus those who seek out outsiders' help (role attribute), those who discuss requirements issues with people abroad (location attribute), and those who go to seniors members to clarify requirements (time working in the project attribute).

RCSN size. RCSN size is the number of members in each RCSN and helps convey the amount of coordination required for each requirement. The proportion of RCT members involved in a particular requirement out of the total members in the project indicates the requirement relative size and suggest how big is the scope of a requirement.

RCSN density. RCSN density is the proportion of ties that exist in the RCSN out of the total possible ties. In requirements-driven collaboration, it is a measurement of how tightly-coupled the requirements-centric team is, and reflects the ability of the team to distribute knowledge about the changes in requirements or clarifications about requirements. For example, a communication-RCSN with high density suggest that the RCT members communicate a lot with others working on the requirement. If seeking clarifications is the topic of discussion in the highly dense communication-RCSN then one may conclude that the requirement is ambiguous and problematic because it necessitates a lot of information exchange to clarify it. Similarly, a communication-RCSN drawn from messages about requirements change that has high density is indicative of a requirement that is highly volatile.

Ties statistics based on RCT members attributes. Another way to reveal patterns of requirements-driven collaboration in a RCSN is to provide statistics about the relationships established by RCT members based on their personal characteristics (attributes). This kind of analysis makes explicit, for example, how many ties are cross-sites or cross-teams characterizing whether collaboration takes place beyond physical and team boundaries.

D.4 Analysis of Communication and Fleeting Knowledge Network Structures in RDC

Network structure – the arrangement of a set of ties linking the actors in the network [111] – is important in the study of requirements-driven collaboration because it allows one to examine patterns of behaviour of those in positions to send information about requirements, or of the entire network making decisions about requirements. I regard a requirements-centric social network as a conduit for propagation of information or the exertion of influence. Each RCT member's place in the overall pattern of relationships, largely determined by personal attributes such as location, experience and role, determines what information that member has access to, and who that member is in a position to influence. Thus, patterns of information flow affect the individual's capabilities in the project and as such there is an important relationship between members' attributes, capabilities and the network structure.

RCSN centralization. RCSN centralization is an expression of how tightly the network is organized around its most central nodes. Network centralization increases as one member (or a few members) have connections to many others while the rest of the RCT is connected to only a few members, thus those in central positions are the ones who most collaborated and potentially acted as project focal points. Degree centrality can reveal who are these members, and contextual information about the project can help understand whether these members are experts on the requirements. Moreover, in requirements-driven collaboration a highly centralized RCSN suggests that knowledge is centralized in few team members, and that the team may be dependent on them to propagate requirements information. This dependency may impact the speed and efficiency that a requirements-centric team discusses and solves requirements-related problems, as well as propagate notifications of changes to requirements.

Core-periphery. The core-periphery test indicates the extent to which the structure of a RCSN 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. In requirements-driven collaboration, the core-periphery test reveals whether the requirements activities turn around a core of RCT members, instead of being centralized in few members like indicated by the network centralization measure. Core members may be those who have most knowledge about the requirements, and that are key for the project progress. A large core indicates that more members are available to propagate requirements information, thus reducing the risk of the team facing collaboration issues because few members are overloaded with work and act as bottlenecks of information exchange. In contrast, peripheral members are to some extent isolated. They may not be receiving the information necessary to complete their requirements-related work. For instance, the test may reveal that newcomers to a project are peripheral in the structure, and their isolation is caused by their lack of familiarity with colleagues and of knowledge on how to work their way inside the team.

Ties reciprocity. Another way to study the structure of a RCSN is to examine where ties, in a directed network, are reciprocal. The higher the index of reciprocal ties, the more stable or equal a RCSN is expected to be is because the members are assumed to mutually exchange information. Moreover, a high reciprocity index sug-

gests a more horizontal structure while the opposite, a low reciprocity index, suggests a more hierarchical structure [66]. For example, the low reciprocity index of a directed awareness-RCSN may be associated with the roles played by the team members and with the restricted interaction channels imposed by a formal organization structure. Project managers may be aware of what testers are working on because the test leader weekly reports the test team progress status, but testers may not have knowledge of what managers are doing. A high reciprocity index in a communication-RCSN suggests that team members seek each others help to develop requirements-related activities and equally discuss requirements-related issues.

Clique. A clique consists of a subset of at least three RCT members of a RCSN in which every possible pair of members is directly connected by a tie and this clique is not contained in any other clique. In requirements-driven collaboration, a clique reveals a set of members who are more closely and intensely tied to one another than they are to other members of the network. In a communication-RCSN, a clique suggests that this set of members are more intensively discussing or negotiating the requirement. It is likely that the members of a clique are the first source of information for each other, thus helping one to understand why certain members are isolated or are less sought by others. Members of a clique may also share working behavior patterns such as they have the preference for the same communication media to discuss a requirement at a distance, thus facilitating collaboration among these members. These members may group together, for example, because of the level of expertise they have about the requirement, because they are physically collocated, or because they play the same role in the project. This measure is of special interest in the study of large requirements-centric social networks that may be found in projects that implement large requirements or that have many dependencies among requirements. A network that contains a high overlap of members in its several cliques suggests a well integrated team, therefore coordination of activities should be facilitated.

D.5 Analysis of Information Flow Patterns in RDC

Each type of RCSN defined in the framework lends itself to the analysis of key actors in the RCSNs. Using the following measures, one can identify patterns of information flow and knowledge sharing within requirements-centric social networks. One can determine whether all members are reachable, whether there are mediators of infor-

mation flow in communication networks, who are the members most aware of what others are doing, among other characteristics of requirements-driven collaboration within the boundaries of individual requirements-centric social networks.

Component. In requirements-driven collaboration it is important that all members receive notification of changes to requirements, for instance. Notification of changes will not reach isolated members. Thus, it is important to know whether or not a RCSN is connected to identify whether information can flow to all members. A RCSN is connected if there is a path between every pair of members in the network. Otherwise the RCSN is disconnected. The members in a disconnected network are partitioned into two or more subsets in which there are no paths between the members in different subsets. The connected subsets in a RCSN are called *components*. Note that the component test differs from the clique test in the sense that the former indicates whether there is a group of members connected to each other (and disconnected from the remaining members) and the latter indicates whether a subset of members is completely connected.

Reachability. A RCT member is reachable by another RCT member if exists any set of ties that connects both members, regardless of how many others fall between them. A communication requirements-centric social network where members cannot be reached may impair the team's ability to share information and engage its members in the requirements-related activities. On the other hand, increasing connectedness in the team may signal increased knowledge sharing and collaboration. One can contrast reachability in different communication networks. For example, there may be a communication path between member A and member B in a network where the topic of discussion is *requirements changes* but A and B may not be connected when it comes to *negotiating requirements*. This difference may indicate that the roles that members play in the project may influence their communication behavior. If some members in a RCSN cannot reach others, there is a potential division in the network. It can also suggest that the team is composed of more than one group. The extent of this division or organization in groups is measured by the *Component* measure.

Cutpoints (or cutsets). In contrast to finding which members are at the core of the network, one can identify which members are weak points in the RCSN and if they are removed along with their connections, the network would become divided

into unconnected parts. If there is one such member, he is called a *cutpoint* [135]. However, if there is a set of actors that fit this criteria, then one would call them a *cutset*. In a communication requirements-centric social network, a team member who is a cutpoint is critical because if he leaves the company or is allocated to work on another project the network will became disconnected and information will not flow between the two disconnected groups, at least until this former member is replaced and his connections are reinstalled. This test reveals how well connected and vulnerable to disruption a requirements-centric social network is.

Degree centrality. Centrality is the extent to which an actor is the centre of a network. Central people tend to have more influence in the network than others. For instance, in a communication-RCSN, a member who is central sends and receives messages to a large number of members in the network. Measures of centrality include degree, betweenness, and closeness centrality. The *degree* centrality indicates the number of ties of a member and is indicative of activity.

Brokerage. Brokerage indicates when an actor, named *broker*, connects two otherwise unconnected actors or subgroups [61]. Brokerage occurs when, in a triad of actors A, B, and C, A has a tie to B, and B has a tie to C, but A has no tie to C. In other words, A needs B to reach C, and therefore B is a broker. The broker actor is in a position to manage or broker information flow. A broker can be problematic if intentionally or unintentionally the actor introduces misunderstandings or limits exchange of information. To calculate brokerage first it is necessary to divide actors into mutually exclusive groups. This partition must be meaningful in accordance to the context of the studied network [10].

To study the flow of information in communication requirements-centric social networks of dependent requirements, one should divide the team members by the requirement that they were assigned to work on. For instance, in the case of a dependency between two requirements, three groups are identified: the group of members who work on the dependent requirement, the group of people who work on the dependee requirement, and those who work on both requirements. Thus, consider that a member s sends information to a receiver member r through a broker member b, and the three requirements group mentioned above. The possible brokerage information flow can be identified: *outgoing flow*, which investigates the presence of brokers who mediate information from the dependee to the dependent requirement; *incoming*

flow, which investigates the presence of brokers who mediate information from the dependent to the dependee requirement; and *consulting flow*, which investigates the presence of brokers who mediate information either from the dependent or from the dependee requirement, and is omitted here to avoid duplication. Information flow brokers are usually the most knowledgeable members of a team regardless of geographical distance [96]. In requirements-driven collaboration, brokers are essential for enabling effective flow of information between RCTs of dependent requirements.

D.6 Analysis of the Alignment among Coordination Needs, Actual Coordination, and Organization Structure in RDC

Because requirements-centric social networks capture communication and work relationships, one can compare different types of networks to learn how similar the collaboration behavior is among the networks, or the effects of one type of relationship on another.

Alignment of RCSNs. The alignment between coordination needs and the teams actual ability to coordinate work (actual social interactions) can be examined by applying the socio-technical congruence (STC) measure [24]. To compute STC one calculates the ratio of actual social interactions over the expected coordination needs informed from the technical dependencies in the project. In the study of requirements-driven collaboration, the assignment- and technical-RCSNs can be used to calculate the coordination needs. Similarly, the communication- and awareness-RCSNs constructed from data on social interaction reflect actual coordination behavior and the team's ability to coordinate, respectively. A STC index can be computed by dividing the number of relationships in a coordination needs RCSN by a social interaction RCSN[1]. In requirements-driven collaboration, a low STC index is a symptom of a larger problem, for example, not coordinating a requirements change with others who work on interrelated requirements or not involving stakeholders owners of dependent features in negotiations of the project scope.

[1]The formal definition of this measure has been presented in Chapter 4 and will not be presented here again to avoid duplication).

In the presence of a formal imposed organization structure that prescribes communication channels between project roles, one can examine the extent to which requirements-driven collaboration patterns are aligned with the organizational structure. To compute such alignment, one should use the Role-based socio-technical congruence measure, which indicates the extent that actual coordination attends coordination needs that are aligned to the formal imposed communication channels. The Role-based socio-technical congruence is calculated in a similar way than in the original Cataldo's congruence measure, however a Role-Role network is considered in the calculation[2].

In addition, the Role-based measure also identifies *false congruence gaps* (false gaps in short), which represent a missing communication between any pair of members who should not directly communicate according to the organization structure. The identification of false gaps is important because they highlight point-to-point direct coordination needs that might not have to be satisfied. Furthermore, the Role-base measure also identifies *backchannel communication*, which is communication that takes place between a pair of members who play roles that should not directly communicate with each other. In addition, backchannel communication properties are also identified, such as who are the roles involved and what topics team members discuss when communicating "behind-the-scenes". Managers can benefit from the awareness that backchannel communication is happening in their projects, and use the detailed information about this type of communication to align or to supplement the formal organization structure.

The Role-based measure has more explanatory power of requirements-driven coordination behavior with respect to influence of roles on amount and type of communication than Cataldo's original measure. In summary, the information revealed by the Role-based measure can be categorized as presented in Figure D.2. For each pair of people in a requirement-centric social network, one can examine the relationship between them and follow the chart to determine the type of coordination activity that connects this pair.

This categorization reveals four different types of relationships between people in a project that allow the classification of coordination activities among team members. These relationships are as follows: *False gap*, where there is no actual coordination but roles were not supposed to talk to each other; *Real gap*, where there is no actual

[2]The formal definition is also not presented here to avoid duplication, thus please refer to Chapter 4.

coordination but coordination needs exist; *Backchannel communication*, where there is actual communication between roles who are not supposed to talk to each other; and *Aligned communication*, where there is actual communication and this communication satisfies coordination needs.

False gaps and backchannel communication can be indicators of whether an organizational restructuring may lead to a better alignment of social and technical coordination. Thus, the Role-based measure can also be used as a tool to assess the health of the organization structure. The scheme presented in Table D.2 can be used to support this assessment. The best-case scenario (1) is if there are a small

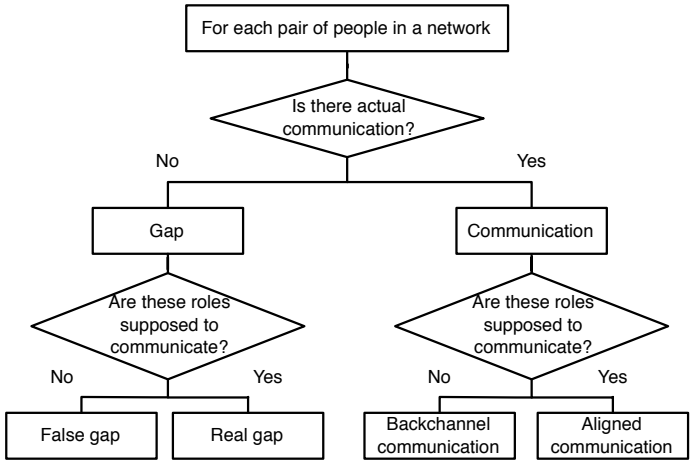


Figure D.2: Flow chart to classify coordination activities

Table D.2: Scheme to assess the organization structure's health

	Few false gaps	Many false gaps
Few backchannel communications	**Nominal situation (1):** Organizational structure aligns with technical dependencies and people are following prescribed communication channels.	**Possible problem situation (4):** people may not be communicating enough.
Many backchannel communications	**Possible problem situation (2):** people are breaking communication structure arbitrarily despite alignment of communication structure with technical dependencies.	**Possible problem situation (3):** poor organizational structure is causing people to compensate using back-channel communication.

number of false gaps as well as a low amount of backchannel communication. This nominal situation means that there is good alignment between the project's technical dependencies and role responsibilities, and that people are obeying the organizational structure; it may also suggest that backchannel communication is not needed due to the good alignment between a project's technical dependencies and organizational structure. A situation that is slightly worse (2) is if there are few false gaps but a lot of backchannel communication—this situation implies that the individuals are not respecting the communication structure despite the fact that the organizational structure is well-aligned. Individuals who are found to constantly break the guidelines may need to be disciplined.

If there is a large number of false gaps, as well as a high amount of backchannel communication (3), this implies that the organizational structure is not working for reasons such as delays in propagating information through the formal channels [57], but that people are compensating through their own means. In this situation, the organizational structure is working against people and thus should be modified. If there is a large number of false gaps, but low backchannel communication (4), then more information about the organization is needed. It would appear that false gaps would get in the way of individual productivity, but there are two reasons that backchannel communication may not actually occur: there is actually no need for it; or individuals are adhering to the communication structure to the detriment of the project.

The identification of the organizational structure's health in light of the coordination behavior has to be assessed with support of contextual information. One needs to consider the influence and impact of organizational processes, policies, and guidelines to decide whether changes on the organizational structure are necessary.

Bibliography

[1] Marilyn Adams, Yvette Tenney, and Richard Pew. Situation Awareness and the Cognitive Management of Complex Systems. *Human Factors*, 37(1):85–104, 1995.

[2] Patricia Adler and Peter Adler. *Handbook of Qualitative Research*, chapter Observational Techniques, pages 377–392. Sage Publications, Thousand Oaks, United States, 1994.

[3] Manju Ahuja and Kathleen Carley. Network Structure in Virtual Organizations. *Organization Science*, 10(6):741–757, November 1999.

[4] Manju Ahuja, Dennis Galleta, and Kathleen Carley. Individual Centrality and Performance in Virtual RD Groups: An Empirical Study. *Management Science*, 49(1):21–38, January 2003.

[5] Ban Al-Ani and David Redmiles. Trust in Distributed Teams: Support through Continuous Coordination. *IEEE Software*, 26(6):35–40, November 2009.

[6] Amer Al-Rawas and Steve Easterbrook. Communication Problems in Requirements Engineering: A Field Study. In *Proceedings of the Conference on Professional Awareness in Software Engineering*, pages 46–60, London, England, February 1996. Royal Society.

[7] Terrance Albrecht and Vickie Ropp. Communicating about Innovation in Networks of Three U.S. Organizations. *Journal of Communication*, 34(3):78–91, September 1984.

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

[9] Chintan Amrit. Coordination in Software Development: The Problem of Task Allocation. In *Proceedings of the Workshop on Human and Social Factors of Software Engineering*, pages 1–7, St. Louis, United States, May 2005. ACM.

[10] Diego Batallas and Ali Yassine. Information Leaders in Product Development Organizational Networks: Social Network Analysis of the Design Structure Matrix. *IEEE Transactions on Engineering Management*, 53(4):570–582, November 2006.

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

[12] Nicolas Bettenburg, Sascha Just, Adrian Schroter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. What Makes a Good Bug Report? In *Proceedings of the International Symposium on Foundations of Software Engineering*, pages 308–318, Atlanta, United States, November 2008. ACM.

[13] Christian Bird, Alex Gourley, Premkumar Devanbu, Michael Gertz, and Anand Swaminathan. Mining Email Social Networks. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 137–143, Shanghai, China, May 2006. ACM.

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

[15] Erin Bradner, Gloria Mark, and Tammie Hertel. Effects of Team Size on Participation, Awareness, and Technology Choice in Geographically Distributed Teams. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, volume 8, pages 271–280, Big Island, Hawaii, January 2003.

[16] Sandra Branco and Rob Williams. *Brazil Culture Smart!: The Essential Guide to Customs and Culture*. Kuperard, London, England, September 2006.

[17] Lars Bratthall and Magne Jørgensen. Can you Trust a Single Data Source Exploratory Software Engineering Case Study? *Empirical Software Engineering*, 7(1):9–26, March 2002.

[18] Frederick Brooks. No Silver Bullet: Essence and Accidents of Software Engineering. *Computer*, 20(4):10–19, April 1987.

[19] Tom Burns and G. Talker. *The Management of Innovation*. Tavistock Publications, London, England, 1961.

[20] Ronald Burt. A Note on Missing Network Data in the General Social Survey. *Social Networks*, 9(1):63–73, March 1987.

[21] Brendan Cain, James Coplien, and Neil Harrison. Social Patterns in Productive Software Development Organizations. *Annals of Software Engineering*, 2(1):259–286, 1996.

[22] Glenn Carroll and Albert Teo. On the Social Networks of Managers. *The Academy of Management Journal*, 39(2):421–440, April 1996.

[23] Marcelo Cataldo and James Herbsleb. Communication Networks in Geographically Distributed Software Development. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 579–588, San Diego, United States, November 2008. ACM.

[24] Marcelo Cataldo, Patrick Wagstrom, James Herbsleb, and Kathleen Carley. Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 353–362, Banff, Canada, November 2006. ACM.

[25] Kathy Charmaz. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. Sage Publications, London, England, January 2006.

[26] Lee Li-Jen Chen and Brian Gaines. A CyberOrganism Model for Awareness in Collaborative Communities on the Internet. *International Journal of Intelligent Systems*, 12(1):31–56, January 1997.

[27] Betty Cheng and Joanne Atlee. Research Directions in Requirements Engineering. In *Proceedings of the Conference on The Future of Software Engineering in conjunction with the International Conference on Software Engineering*, pages 285–303, Minneapolis, United States, May 2007. ACM.

[28] J. Cleland-Huang, R. Settimi, Xuchang Zou, and P. Solc. The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. In *Proceedings of the International Requirements Engineering Conference*,

pages 39–48, Minneapolis, United States, September 2006. IEEE Computer Society.

[29] David Conrath, Christopher Higgins, and Ronald McClean. A Comparison of the Reliability of Questionnaire Versus Diary Data. *Social Networks*, 5(3):315–322, September 1983.

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

[31] Jane Coughlan and Robert Macredie. Effective Communication in Requirements Elicitation: A Comparison of Methodologies. *Requirements Engineering*, 7(2):47–60, 2002.

[32] John Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications, Los Angeles, Unites States, 2nd edition, July 2002.

[33] Rob Cross, Stephen Borgatti, and Andrew Parker. Making Invisible Work Visible: Using Social Network Analysis to Support Strategic Collaboration. *California Management Review*, 44(2):25–46, January 2002.

[34] Rob Cross and Andrew Parker. *The Hidden Power of Social Networks: Understanding How work Really Gets Done in Organizations*. Harvard Business School Press, Boston, United States, June 2004.

[35] Kevin Crowston, Kangning Wei, Qing Li, and James Howison. Core and Periphery in Free/Libre and Open Source Software Team Communications. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, Kauai, United States, January 2006. IEEE Computer Society.

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

[37] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A Diary Study of Task Switching and Interruptions. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 175–182, Vienna, Austria, April 2004. ACM.

[38] Asa Dahlstedt and Anne 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.

[39] Daniela Damian, Luis Izquierdo, Janice Singer, and Irwin Kwan. Awareness in the Wild: Why Communication Breakdowns Occur. In *Proceedings of the International Conference on Global Software Engineering*, pages 81–90, Munich, Germany, August 2007. IEEE Computer Society.

[40] Daniela Damian and Didar Zowghi. RE Challenges in Multi-Site Software Development Organisations. *Requirements Engineering*, 8(3):149–160, August 2003.

[41] Alan Davis. *Software Requirements: Objects, Functions and States*. Prentice Hall, New Jersey, United States, 2nd edition, 1993.

[42] Cleidson de Souza, Jon Froehlich, and Paul Dourish. Seeking the Source: Software Source Code as a Social and Technical Artifact. In Mark Pendergast, Kjeld Schmidt, Gloria Mark, and Mark Ackerman, editors, *Proceedings of the International Conference on Supporting Group Work*, pages 197–206, Sanibel Island, United States, November 2005. ACM.

[43] Cleidson de Souza and David Redmiles. The Awareness Network: To Whom Should I Display My Actions? And, Whose Actions Should I Monitor? In *Proceedings of the European Conference on Computer-Supported Cooperative Work*, pages 99–117, Limerick, Ireland, September 2007. Springer.

[44] Cleidson de Souza, David Redmiles, Li-Te Cheng, David Millen, and John Patterson. Sometimes You Need to See Through Walls - A Field Study of Application Programming Interfaces. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 63–71, Chicago, United States, November 2004. ACM.

[45] Paul DiGangi and Molly Wasko. Would You Share? Examining How Knowledge Type and Communication Channel Influence Knowledge Sharing. In *Proceedings of the Americas Conference on Information Systems*, Toronto, Canada, August 2008. AIS.

[46] Paul Dourish and Victoria Bellotti. Awareness and Coordination in Shared Workspace. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 104–114, Toronto, Canada, October 1992. ACM.

[47] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. *Guide to Advanced Empirical Software Engineering*, chapter Selecting Empirical Methods for Software Engineering Research, pages 285–311. Number 11. Springer-Verlag, London, England, November 2008.

[48] Kate Ehrlich and Klarissa Chang. Leveraging Expertise in Global Software Teams: Going Outside Boundaries. In *Proceedings of the International Conference on Global Software Engineering*, pages 149–158, Florianópolis, Brazil, October 2006. IEEE Computer Society.

[49] Kate Ehrlich, Mary Helander, Giuseppe Valetto, Stephen Davies, and Clay Williams. An Analysis of Congruence Gaps and Their Effect on Distributed Software Development. In *Workshop on Socio-Technical Congruence, in conjunction with the International Conference on Software Engineering*, Leipzig, Germany, May 2008. ACM.

[50] K. El Emam and N. Madhavji. A Field Study of Requirements Engineering Practice in Information Systems Development. In *Proceedings of the International Symposium on Requirements Engineering*, pages 68–80, York, England, March 1995. IEEE Computer Society.

[51] Hakan Erdogmus, Maurizio Morisio, and Marco Torchiano. On the Effectiveness of the Test-First Approach to Programming. *IEEE Transactions on Software Engineering*, 31(3):226–237, March 2005.

[52] Alberto Espinosa, Sandra Slaughter, Robert E. Kraut, and James Herbsleb. Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*, 24(1):135–169, 2007.

[53] Lucia Falzon. Determining Groups from the Clique Structure in Large Social Networks. *Social Networks*, 22(2):159–172, May 2000.

[54] Samer Faraj and Lee Sproull. Coordinating Expertise in Software Development Teams. *Management Science*, 46(12):1554–1568, December 2000.

[55] Dalmar Fisher. *Communication in Organizations*. West Publishing Company, Minneapolis, United States, 2nd edition, January 1993.

[56] Bent Flyvbjerg. Five Misunderstandings about Case-Study Research. *Qualitative Inquiry*, 12(2):219–245, April 2006.

[57] Allan Frank and Judi Brownell. *Organizational Communication and Behavior: Communicating to Improve Performance, 2+2=5*. Harcourt Brace College Publishers, New York, United States, 2nd edition, December 1989.

[58] Linton Freeman. Centrality in Social Networks: Conceptual Clarification. *Social Networks*, 1(3):215–239, 1978/1979.

[59] Linton Freeman, Douglas Roeder, and Robert Mulholland. Centrality in Social Networks: II. Experimental Results. *Social Networks*, 2(2):119–141, 1979/1980.

[60] Barney Glaser and Anselm Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, 1967.

[61] Roger Gould and Roberto Fernandez. Structures of Mediation: A Formal Approach to Brokerage in Transaction Networks. *Sociological Methodology*, 19:89–126, 1989.

[62] Rebecca Grinter and Margery Eldridge. y do tngrs luv 2 txt msg? In *Proceedings of the Seventh European Conference on Computer-Supported Cooperative Work*, pages 219–238, Bonn, Germany, September 2001. Kluwer Academic Publishers.

[63] The Standish Group. Chaos Report. Report, The Standish Group, 1995.

[64] Carl Gutwin and Saul Greenberg. *Team Cognition: Process and Performance at the INter- and Intra-Individual Level*, chapter The Importance of Awareness for Team Cognition in Distributed Collaboration, pages 177–201. APA Press, 2004.

[65] Carl Gutwin, Mark Roseman, and Saul Greenberg. A Usability Study of Awareness Widgets in a Shared Workspace Groupware System. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 258–267, Boston, United States, 1996. ACM.

[66] Robert Hanneman and Mark Riddle. *Introduction to Social Network Methods*. University of California, Riverside, United States, 2005.

[67] Jurgen Hauschildt and Gerhard Schewe. Gatekeeper and Process Promoter: Key Persons in Agile Innovative Organizations. *International Journal of Agile Management Systems*, 2(2):96–103, 2000.

[68] James Herbsleb and Audris Mockus. An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 29(6):481–494, June 2003.

[69] James Herbsleb and Deependra Moitra. Global Software Development. *IEEE Software*, 18(2):16–20, March 2001.

[70] Pamela Hinds and Sara Kiesler. Communication Across Boundaries: Work, Structure, and Use of Communication Technologies in a Large Organization. *Organization Science*, 6(4):373–393, July 1995.

[71] Pamela Hinds and Cathleen McGrath. Structures that Work: Social Structure, Work Structure and Coordination Ease in Geographically Distributed Teams. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 343–352, Banff, Canada, November 2006. ACM.

[72] Liaquat Hossain, Andre Wu, and Kennetg Chung. Actor Centrality Correlates to Project Based Coordination. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 363–372, Banff, Canada, November 2006. ACM.

[73] James Howison, Keisuke Inoue, and Kevin Crowston. Social Dynamis of Free and Open Source Team Communications. In *Proceedings of the International Conference on Open Source Software*, pages 319–330, Lake Como, Italy, June 2006. Springer.

[74] Susan Jackson. *Group Process and Productivity*, chapter Team Composition in Organizational Settings: Issues in Managing an Increasingly Diverse Workforce, pages 138–173. Number 6. Sage Publications, Newburry, United States, 1992.

[75] Andreas Jedlitschka, Marcus Ciolkowski, and Dietmar Pfahl. *Guide to Advanced Empirical Software Engineering*, chapter Reporting Experiments in Software

Engineering, pages 201–228. Number 8. Springer, London, England, November 2008.

[76] Amela Karahasanovic, Unni Nyhamar Hinkel, Dag Sjoberg, and Richard Thomas. Comparing of Feedback-Collection and Think-Aloud Methods in Program Comprehension Studies. *Behaviour and Information Technology*, 28(2):139–164, March 2009.

[77] Sara Kiesler and Jonathon Cummings. *Distributed Work*, chapter What Do We Know about Proximity and Distance in Work Groups? A Legacy of Research, pages 57–80. Number 3. MIT Press, Boston, United States, May 2002.

[78] Barbara Kitchenham and Shari Lawrence Pfleeger. Principles of Survey Research Part 4: Questionnaire Evaluation. *ACM SIGSOFT Software Engineering Notes*, 27(3):20–23, May 2002.

[79] Richard Klimoski and Susan Mohammed. Team Mental Model: Construct or Metaphor? *Journal of Management*, 20(2):403–437, April 1994.

[80] Alden Klovdahl. A Note on Images of Networks. *Social Networks*, 3(3):197–214, 1981.

[81] Julia Kotlarsky, Paul van Fenema, and Leslie Willcocks. Developing a Knowledge-Based Perspective on Coordination: The Case of Global Software Projects. *Information and Management*, 45(2):96–108, 2008.

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

[83] David Krackhardt and Robertt Stern. Informal Networks and Organizational Crises: An Experiemental Simulation. *Social Psychology Quarterly*, 51(2):123–140, June 1988.

[84] Robert Kraut and Lynn Streeter. Coordination in Software Development. *Communications of the ACM*, 38(3):69–81, March 1995.

[85] Irwin Kwan, Adrian Schroter, and Daniela Damian. A Weighted Congruence Measure. In *Workshop on Socio-Technical Congruence, in conjunction with the International Conference on Software Engineering*, pages 32–35, Vancouver, Canada, May 2009. ACM.

[86] Danielle De Lange, Filip Agneessens, and Hans Waege. Asking Social Network Questions: A Quality Assessment of Different Measures. *Social Methodologies*, 1(2):351–378, 2004.

[87] Paul Lawrence and Jay Lorsch. *Organization and Environment*. Harvard Business School Press, Boston, United States, 1967.

[88] Soo Ling Lim, Daniele Quercia, and Anthony Finkelstein. StakeNet: Using Social Networks to Analyse the Stakeholders of Large-Scale Software Projects. In *Proceedings of the International Conference on Software Engineering*, pages 295–304, Cape Town, South Africa, May 2010. ACM.

[89] Soo Ling Lim, Daniele Quercia, and Anthony Finkelstein. StakeSource: Harnessing the Power of Crowdsourcing and Social Networks in Stakeholder Analysis. In *Proceedings of the International Conference on Software Engineering*, pages 239–242, Cape Town, South Africa, May 2010. ACM.

[90] Yuan Long and Keng Siau. Social Network Structures in Open Source Software Development Teams. *Journal of Database Management*, 18(2):25–40, April 2007.

[91] Yuan Long and Keng Siau. Impacts of Social Network Structure on Knowledge Sharing in Open Source Software Development Teams. In *Proceedings of the Americas Conference on Information Systems*, Toronto, Canada, August 2008. AIS.

[92] Linda Macaulay. *Requirements Engineering*. Springer-Verlag, New York, United States, April 1996.

[93] Alan MacCormack and John Rusnak anbd Carliss Baldwin. Exploring the Duality between Product and Organizational Architectures: A Test of the Mirroring Hypothesis. Working Paper 08-039, Harvard Business School, Boston, United States, October 2008.

[94] Nazim Madhavji. The Prism Model of Changes. In *Proceedings of the Thirteenth International Conference on Software Engineering*, pages 166–177, Austin, United States, May 1991. IEEE Computer Society.

[95] Thomas Malone and Kevin Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1):87–119, March 1994.

[96] Sabrina Marczak, Daniela Damian, Ulrike Stege, and Adrian 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.

[97] Sabrina Marczak, Irwin Kwan, and Daniela Damian. Investigating Collaboration Driven by Requirements in Cross-Functional Teams. In *Collaboration and Intercultural Issues on Requirements Communication, Understanding, and Soft-skills Workshop, in conjunction with International Requirements Engineering Conference*, Atlanta, United States, September 2009. IEEE Computer Society.

[98] Allen Milewski, Marilyn Tremaine, Felix Kobler, Richard Egan, Siling Zhang, and Patrick O'Sullivan. Guidelines for Effective Bridging in Global Software Engineering. *Software Process Improvement and Practice*, 13:477–492, October 2008.

[99] J. Clyde Mitchell. *Social Networks in Urban Situations: Analyses of Personal Relationships in Central African Towns*. Manchester University Press, Manchester, United Kingdom, November 1969.

[100] Jacob Moreno. *Who Shall Survive? A New Approach to the Problem of Human Interrelations*. Nervous and Mental Disease Monograph. Nervous and Mental Disease Publising, Washington, United States, 1934.

[101] Bashar Nuseibeh and Steve Easterbrook. Requirements Engineering: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering in conjunction with the International Conference on Software Engineering*, pages 35–46, Limerick, Ireland, June 2000. ACM.

[102] Briony J Oates. *Researching Information Systems and Computing*. Sage Publications, London, England, 2006.

[103] Masao Ohira, Naoki Ohsugi, Tetsuya Ohoka, and Ken ichi Matsumoto. Accelerating Cross-Project Knowledge Collaboration Using Collaborative Filtering and Social Networks. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 111–115, St. Louis, United States, May 2005. ACM.

[104] David Parnas. Software Engineering Programmers are not Computer Science Programmers. *Annals of Software Engineering*, 6(1):19–37, 1999.

[105] Geoff Payne and Judy Payne. *Key Concepts in Social Research*. Sage Publications, London, England, 2004.

[106] Dewayne Perry, Adam Porter, and Lawrence Votta. Empirical Studies of Software Engineering: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, pages 345–355, Limerick, Ireland, June 2000. ACM.

[107] Dewayne Perry, Nancy Staudenmayer, and Lawrence Votta. People, Organizations, and Process Improvement. *IEEE Software*, 11(4):36–45, July 1994.

[108] Colin Potts and Lara Catledge. Collaborative Conceptual Design: A Large Software Project Case Study. *Computer Supported Cooperative Work Journal*, 5(4):415–445, 1996.

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

[110] Ramachandra Rao and Suraj Bandyopadhyay. Measures of Reciprocity in a Social Network. *Sankhyā: The Indian Journal of Statistics, Series A*, 49(2):141–188, June 1987.

[111] Everett Rogers and Lawrence Kincaid. *Communication Networks: Toward a New Paradigm for Research*. Free Press, New York, United States, June 1980.

[112] Christian Del Rosso. Comprehend and Analyze Knowledge Networks to Improve Software Evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, 21:189–215, April 2009.

[113] Diane Rulke and Joseph Galaskiewicz. Distribution of Knowledge, Group Network Structure, and Group Performance. *Management Science*, 46(5):612–625, May 2000.

[114] Roberto Hernández Sampieri, Carlos Fernández-Collado, and Pilar Baptista Lucio. *Metodología de la Investigación*. Mc Graw Hill, Ciudad de México, México, cuarta edition, 1997.

[115] Anita Sarma and James Herbsleb. Using Development Experience to Calculate Congruence. In *Workshop on Socio-Technical Congruence, in conjunction with the International Conference on Software Engineering*, Leipzig, Germany, May 2008. ACM.

[116] Anita Sarma, James Herbsleb, and André van der Hoek. Challenges in Measuring, Understanding, and Achieving Socio-Technical Congruence. In *Workshop on Socio-Technical Congruence, in conjunction with the International Conference on Software Engineering*, Leipzig, Germany, May 2008. ACM.

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

[118] Helen Sharp and Hugh Robinson. An Ethnographic Study of XP Practice. *Empirical Software Engineering*, 9(4):353–375, 2004.

[119] Helen Sharp and Hugh Robinson. *Agile Software Development: Current Research and Future Directions*, chapter Three 'C's of Agile Practice: Collaboration, Co-ordination and Communication, pages 61–86. Number 4. Springer-Verlag, London, England, 2010.

[120] Janice Singer, Susan Sim, and Timothy Lethbridge. *Guide to Advanced Empirical Software Engineering*, chapter Software Engineering Data Collection for Field Studies, pages 9–34. Number 1. Springer-Verlag, London, England, November 2008.

[121] Iam Sommerville and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. Willey, New York, United States, 1997.

[122] Raymond Sparrowe, Robert Liden, and Maria Kraimer. Social Networks and the Performance of Individuals and Groups. *Academy of Management Journal*, 44(2):316–325, April 2001.

[123] Robert Stake. *The Art of Case Study Research*. Sage Publications, Thousand Oaks, United States, 1995.

[124] Ivan Steiner. *Group Process and Productivity*. Academic Press Inc, New York, United States, September 1972.

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

[126] Anselm Strauss and Juliet Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage Publications, Thousand Oaks, United States, 2nd edition, 1998.

[127] Markus Strohmaier, Michel Wermelinger, and Yijun Yu. Using Network Properties to Study Congruence of Software Dependencies and Maintenance Activities in Eclipse. In *Workshop on Socio-Technical Congruence, in conjunction with the International Conference on Software Engineering*, pages 63–66, Vancouver, Canada, May 2009. ACM.

[128] James Tam and Saul Greenberg. A Framework for Asynchronous Change Awareness in Collaborative Documents and Workspaces. *International Journal of Human-Computer Studies*, 64:583–598, 2006.

[129] Wenpin Tsai. Social Structure of "Coopetition" Within a Multiunit Organization: Coordination, Competition, and Intraorganizational Knowledge Sharing. *Organization Science*, 13(2):179–190, March 2002.

[130] Giuseppe Valetto, Mary Helander, Kate Ehrlich, Sunita Chulani, Mark Wegman, and Clay Williams. Using Software Repositories to Investigate Sociotechnical Congruence in Development Projects. In *Proceedings of the International Workshop on Mining Software Repositories, in conjunction with the International Conference on Software Engineering*, Minneapolis, United States, May 2007. IEEE Computer Society.

[131] Jurriaan van Reijsen and Remko Helms. Revealing Knowledge Networks from Computer Mediated Communication in Organizations. In *Proceedings of the European Conference on Information Systems*, pages 2503–2515, Verona, Italy, June 2009.

[132] Andrew Van De Ven, Andre Delbecq, and Richard Koenig Jr. Determinants of Coordination Modes within Organizations. *American Sociological Review*, 41(2):322–338, April 1976.

[133] Patrick Wagstrom and James Herbsleb. Individualized Socio-Technical Congruence. In *Workshop on Socio-Technical Congruence, in conjunction with the International Conference on Software Engineering*, Leipzig, Germany, May 2008. ACM.

[134] Patrick Wagstrom, James Herbsleb, and Kathleen Carley. Decaying Socio-Technical Congruence as a Method to Account for Architectural Changes. In *Workshop on Socio-Technical Congruence, in conjunction with the International Conference on Software Engineering*, pages 71–78, Vancouver, Canada, May 2009. ACM.

[135] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Crambidge University Press, Crambidge, United Kingdom, 1994.

[136] Jim Whitehead. Collaboration in Software Engineering: A Roadmap. In Lionel Briand and Alexander Wolf, editors, *Proceedings of the Workshop on the Future of Software Engineering, in conjunction with the International Conference on Software Engineering*, pages 214–225, Minneapolis, United States, May 2007. ACM.

[137] Timo Wolf, Thanh Nguyen, and Daniela Damian. Does Distance Still Matter? *Software Process Improvement and Practice*, 13(6):493–510, November 2008.

[138] Timo Wolf, Adrian Schroter, Daniela Damian, and Thanh Nguyen. Predicting Failures using Social Network Analysis on Developer Communication. In *Proceedings of the International Conference on Software Engineering*, pages 1–11, Vancouver, Canada, May 2009. IEEE Computer Society.

[139] Timo Wolf, Adrian Schroter, Daniela Damian, Lucas Panjer, and Thanh Nguyen. Mining Task-Based Social Networks to Explore Collaboration in Software Teams. *IEEE Software*, 26(1):58–66, January 2009.

[140] Robert Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods Series. Sage Publications, Thousand Oaks, United States, second edition, 1994.