The Study of Socio-technical Coordination Using A Socio-technical Congruence Model

by

Irwin Hin-Bong Kwan
B.A.Sc., University of Ottawa, 2002
M.Math., University of Waterloo, 2005

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

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

The Study of Socio-technical Coordination Using A Socio-technical Congruence Model

by

Irwin Hin-Bong Kwan
B.A.Sc., University of Ottawa, 2002
M.Math., University of Waterloo, 2005

Supervisory Committee

Dr. D. Damian, Supervisor
(Department of Computer Science, University of Victoria)

Dr. M. A. Storey, Departmental Member
(Department of Computer Science, University of Victoria)

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

Dr. A. Hadwin, Outside Member
(Department of Educational Psychology and Leadership Studies, University of Victoria)

Supervisory Committee

Dr. D. Damian, Supervisor
(Department of Computer Science, University of Victoria)

Dr. M. A. Storey, Departmental Member
(Department of Computer Science, University of Victoria)

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

Dr. A. Hadwin, Outside Member
(Department of Educational Psychology and Leadership Studies, University of Victoria)

ABSTRACT

Coordination in software development, especially in global software development, is important because a team cannot perform well unless its team members communicate and maintain awareness of each other's activities. In order to improve **socio-technical coordination**, which is coordination among team members who work on interdependent technical entities, it must be conceptualized and measured. One measurement of coordination is **socio-technical congruence**, which calculates the alignment between technical relationships and social relationships.

The problem is that there are a large number of social and technical factors to consider when using socio-technical congruence to study coordination. Current limitations with socio-technical congruence include the inability to represent the size of *gaps* in coordination between people, the sparse understanding of the role of *awareness* in conjunction with other coordination mechanisms, and the lack of a technique with which to model people who are involved in certain *communication patterns*, but not assigned to technical tasks.

To address these limitations, this dissertation describes a socio-technical congruence model to study socio-technical coordination. The model focuses on refining conceptualizations of technical and social relationships between people, on describing an improved gap technique for calculating socio-technical alignment, and on providing guidelines on how to study coordination in teams using the socio-technical congruence model. I first develop the model theoretically from related work. I then conduct two empirical investigations to address limitations of the model. The first study examines

awareness, using observational studies, in a small global team. The second study examines *important communicators* and *people who emerge in coordination* despite having no technical relationships by examining email archives from the same team. I conduct a third empirical investigation of a large global team to apply the model to study the *relationship between socio-technical congruence and team performance* with the project's repository. Finally, I revisit the model and improve it based on the empirical findings.

The model refines conceptualizations of relationships, classifies emergent people who are suddenly involved with a task or a team during the project, and represents multi-variable relationships. It includes a template and an accompanying process for applying socio-technical congruence to study socio-technical coordination. This model enables researchers to study socio-technical coordination and analyze its effect on software engineering outcomes such as performance and quality.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

Thanks first to my parents, especially my father who insisted and persistently encouraged me not to give up, but also directing me to finish. This adherence to discipline and self-improvement is what I remember from them when I was young. This attitude has been instilled in me at a young age and their continual reminders and encouragement has allowed me to persist through the years of doing a Ph.D.

I would like to thank my supervisor, Daniela Damian, who from me crafted a researcher. We have both learned a lot in these years and your care and guidance has made me into a quick, critical thinker able to adapt to new situations.

There are a number of colleagues who have helped me throughout the years, but the two that stand out are Sabrina Marczak and Adrian Schröter. Thanks Sabrina for the advice and the support, and for being that person to go to. Your thoughtfulness and kindness made the journey possible. Thank you Adrian for inspiration, for motivation, and for insight. Not only are you bright, but you are willing to share that intelligence with the world.

Thanks to my committee members Brian Gaines, Peggy Storey, Ally Hadwin, as well as my external Jane Huang, who were willing to support and endorse my research, and to lend to it scrutinizing eyes. You are the quality assurance specialists, the ones who ensure that my work is something I can be proud of.

Thanks to Dan Berry for his ongoing advice and support on all things.

Thanks to the following colleagues that I have encountered throughout my many years in the group: Chris Hanlon, Luis Izquierdo, Kedar Shrikhande, Florian Huber, Andrew Swerdlow, Lucas Panjer, Fabio Calefato, Thanh Nguyen, Rafael Prikladniki, Timo Wolf, Neil Burroughs, Jorge Aranda, Indi Nurdiani, and Germán Póo-Caamaño. Thanks also to all of the great people on `#beer` for their understanding of the Ph.D process. Abram Hindle, Dave Evans, Mike Patterson, and Rob Warren deserve their shout outs.

This dissertation was made possible with funding from the University of Victoria and NSERC.

DEDICATION

To Ellie.

Chapter 1

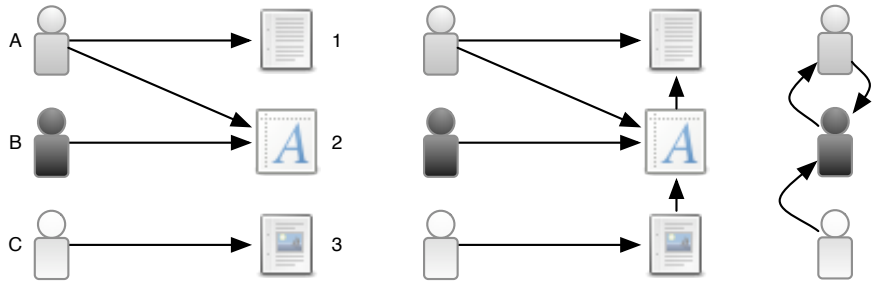
Introduction

Coordinating the efforts of individuals in a team is necessary to build modern software systems [133, 112] because these systems have high complexity [64, 218], extensive interdependencies between components [90, 191, 77], and a need for domain-specific knowledge [218]. Global software development further increases the difficulty of maintaining such coordination [111, 119, 45] because the distance reduces the awareness between team members [72, 89]. Socio-technical coordination is useful for examining global software teams because such teams are increasingly common [31, 112, 179] and must cope with interesting social dynamics [119, 156]. Communication in global software development is often recorded in artifacts which provide visibility of communication and the opportunity to capture and represent social relationships.

Socio-technical coordination, which is the coordination that occurs in an environment where social and technical factors interact with each other, is an important influence on software project outcomes [90, 64]. A good alignment between social and technical factors is believed to have a generally-positive effect on the performance of software teams [50, 89]. To study the effect of aligned coordination on software-engineering outcomes such as quality or cost, a method is needed to conceptualize and represent these factors in a way that captures the context of an organization. **Socio-technical congruence**, a measure proposed by Cataldo et al. [50], is one such method that can calculate the "fit" between a team's interpersonal social relationships and work-derived technical relationships.

1.1 Socio-technical Congruence Example

Figure 1.1 shows an example of three software developers working on three artifacts that are dependent on each other. In the task assignment step, the team members are each assigned to work on a task (Figure 1.1(a)). Some of these tasks have dependencies on other tasks (Figure 1.1(b)). Because the tasks have dependencies, it implies that the people who work on these tasks have dependencies on each other as well, and therefore should communicate (Figure 1.1(c)). As Task 3 depends on Task 2, which depends on Task 1, it implies that Person C should communicate with Person B.



(a) Task Assignment: Individuals Assigned to Work on a Task

(b) Task Dependencies: Tasks Depend on Each Other

(c) Technical Relationships: the Individuals Should Coordinate According to their Task Dependencies

Figure 1.1: An Example Demonstrating the Application of Socio-technical Congruence

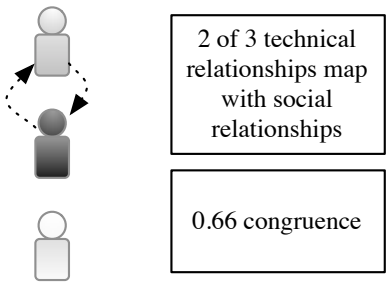


Figure 1.2: Calculating Congruence by Aligning Social Relationships and Technical Relationships

Person A and Person B should communicate with each other because Person A is assigned to both Task 1 and Task 2. A relationship between two team members based on their work is called a **technical relationship**.

Once the technical relationships are calculated, we can align a team member's technical relationships with his coordination activities in practice, referred to as **social relationships**. For example, in Figure 1.2, if A and B talk with each other, but not with C, then only two of the technical relationships are satisfied and we have $\frac{2}{3} = 0.\bar{6}$ congruence.

If two individuals have a technical relationship, but do not coordinate, then there is a **gap** between these two individuals. A gap suggests the existence of a coordination problem. One of the goals of socio-technical congruence is to minimize the number of gaps, either by maintaining good coordination between individuals who have a technical relationship, or by reducing the number of technical dependencies in the project and therefore reducing the number of technical relationships [187].

1.2 Studying Socio-technical Coordination

Many social interactions affect socio-technical coordination in global software development. Socio-technical coordination has been studied using explicit communication as the main interaction between team members [50, 30, 101, 84, 47, 166, 223]. These studies of coordination in software engineering place a large emphasis on communication, usually conceptualizing it as email, mailing lists, or issue-tracking comments, or inquiring about communication behaviour through questionnaires. However, there are a number of social situations that are not well-conceptualized.

The Influence of Awareness on Socio-technical Coordination

The traces that these people leave on their environments provide important cues to others about their availability and their presence, and this knowledge of others affects coordination [89]. **Awareness** represents ongoing knowledge of other's actions and their influence on one's own actions. Awareness is "having an understanding of the activities of others that form a context for your own activities" [80]. Existing work has identified that team members who are familiar with each other collaborate even after they are no longer on the same project [85], and that awareness appears to benefit coordination [89]. In addition, distance reduces the awareness that team members have of each other [72] and that too much communication can lead to information overload [69]. In some cases, communication is not even necessary for coordination [76, 34]. Though awareness has been examined in the context of global software teams, it is especially difficult to conceptualize when applying socio-technical congruence.

Communicators Unrelated to Technical Artifacts

The technical relationships in socio-technical congruence focus heavily on code artifacts [50, 84, 188]. Even in a case where requirements are used instead of code, the project manager could not be included in the conceptualization [151]. Many people who are not technically involved in the project extensively communicate with others about work. **Explicit communication** is defined as intentionally sending information to others [175, 133]. Who is involved in communication reflects directly on a team's ability to coordinate. Identifying communication patterns may highlight people who should be involved in collaborative work, but are not included in technical relationship networks. Two patterns of interest are **important communicators**, who are people that are heavily involved in work-related communication throughout the team, and **emergent people**, who are people included in an online discussion only after an initial set of recipients has been contacted [72, 69].

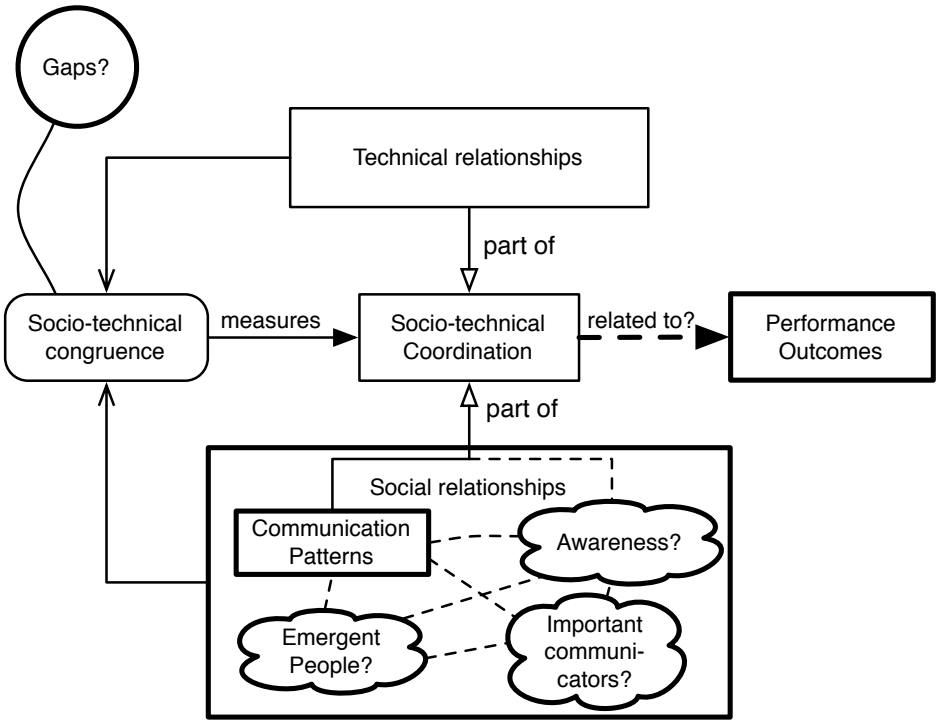


Figure 1.3: Aspects of Socio-technical Coordination

The Applicability of Socio-technical Congruence

A small number of studies show a positive correlation between socio-technical congruence and various outcomes [50, 48, 84, 101]. The socio-technical congruence measurement has promise, but continuing to establish the relationship between socio-technical congruence and the performance of a software-development team is important to discover the limits of applicability of the measure. More studies of the relationship between socio-technical congruence and coordination-intensive team activities are needed to firmly establish the conditions under which socio-technical congruence is beneficial to software engineering outcomes.

Social Aspects in Socio-technical Coordination in this Dissertation

The relationship between the aspects of coordination examined in this dissertation are mapped in Figure 1.3. Socio-technical coordination is made up of technical relationships and social relationships. Socio-technical congruence takes elements from conceptualizations of both relationships and measures socio-technical coordination. Communication is the most common way of conceptualizing social interactions in socio-technical coordination. Other social relationships, such as awareness, emergent people, and important communicators, are not as clearly understood. Gaps are also an area in socio-technical congruence that requires further investigation.

1.3 Problem Statement

Socio-technical coordination is important to study, so a method such as socio-technical congruence is needed to conceptualize it. However, socio-technical congruence has a number of limitations when it is used to study socio-technical coordination in software teams.

A gap in socio-technical congruence is represented as a binary relationship. The socio-technical congruence measure identifies when people's technical relationships are satisfied by their social relationships. If the technical relationship is not satisfied with a social relationship, the resulting situation is described as a **gap**. The current socio-technical congruence measurement considers that one social relationship is sufficient to cover every technical relationship—there is no conceptualization that represents multiple relationships between the two same individuals, nor is there a way to identify if two individuals' coordination is slightly out of alignment with their technical work compared with being not aligned at all. There are likely different "sizes" of gaps that are affected by the work and by the team's communication. Without this fine-grained representation of gaps, it is difficult to identify which individuals in the network have important relationships that must be bridged with social interactions.

Socio-technical congruence studies are difficult to compare. Another limitation of socio-technical congruence is that studies are difficult to compare because of the large number of relationships and entities involved. Many existing studies use different conceptualizations for technical and social relationships, ranging from code changes [84] to co-changed files [50]. Without a way to compare these conceptualizations, it is tiring to catalog the effect of socio-technical congruence on software-engineering outcomes.

The influence of awareness on a team's technical work, in the context of socio-technical congruence, is not well-understood. Most socio-technical congruence studies extract technical and communication data from repositories, which has been shown as revealing only a small part of the story [18]. There are numerous social and organizational factors, such as awareness, team behaviour, and roles, that influence coordination [196, 69, 107, 89]. If awareness is not represented in socio-technical congruence, then many aspects of coordination that have been shown as affecting team behaviour will not be included and socio-technical congruence would not be able to accurately represent social interaction between team members in a software team.

Current conceptualizations of socio-technical congruence do not always involve every relevant team member. Some patterns of communication may indicate exceptional situations that involve team members who are not necessarily

part of the team or the current task. Currently, there are no socio-technical congruence techniques that incorporate explicitly important communicators within a global software team [151]—an **important communicator** is someone who is heavily involved in work-related communication sent throughout a team. Another communication pattern that is not explored is **emergent people**, or people who are involved in a conversation after it is initiated [69, 72].

If these situations are not considered, then people who are relevant to a global software team may not be represented in a socio-technical congruence model.

1.4 Research Goal

The goal of this research is to develop a socio-technical congruence model that can be used for the study of socio-technical coordination in global software teams. Specifically, the model addresses the limitations with using socio-technical congruence to study socio-technical coordination by describing conceptualizations of social and technical relationships, techniques to calculate alignment, and instructions on how to handle exceptional situations. This research refines the conceptualizations of relationships in socio-technical congruence, improves the representation of alignment, and describe guidelines and an approach to apply socio-technical congruence to study socio-technical coordination.

1.5 Research Methodology

To accomplish my research goal, I used the following methodology (Figure 1.4).

1. I developed a theoretical model of socio-technical congruence for the study of socio-technical coordination by examining related work. During this process, I attempted to address the limitations in Section 1.3, but two limitations were still outstanding: the influence of awareness on socio-technical congruence was unclear, and team members involved in the project were not always captured in socio-technical congruence representations. These limitations were addressed in the following empirical studies. (Chapter 3).

2. I performed an empirical investigation to examine the limitation regarding awareness in the socio-technical congruence model (Chapter 4).

3. I performed an empirical investigation to examine the limitation regarding important and emergent people in socio-technical congruence model (Chapter 5).

4. I performed an empirical study in which I applied the socio-technical congruence model to investigate the relationship between socio-technical congruence and team performance (Chapter 6).

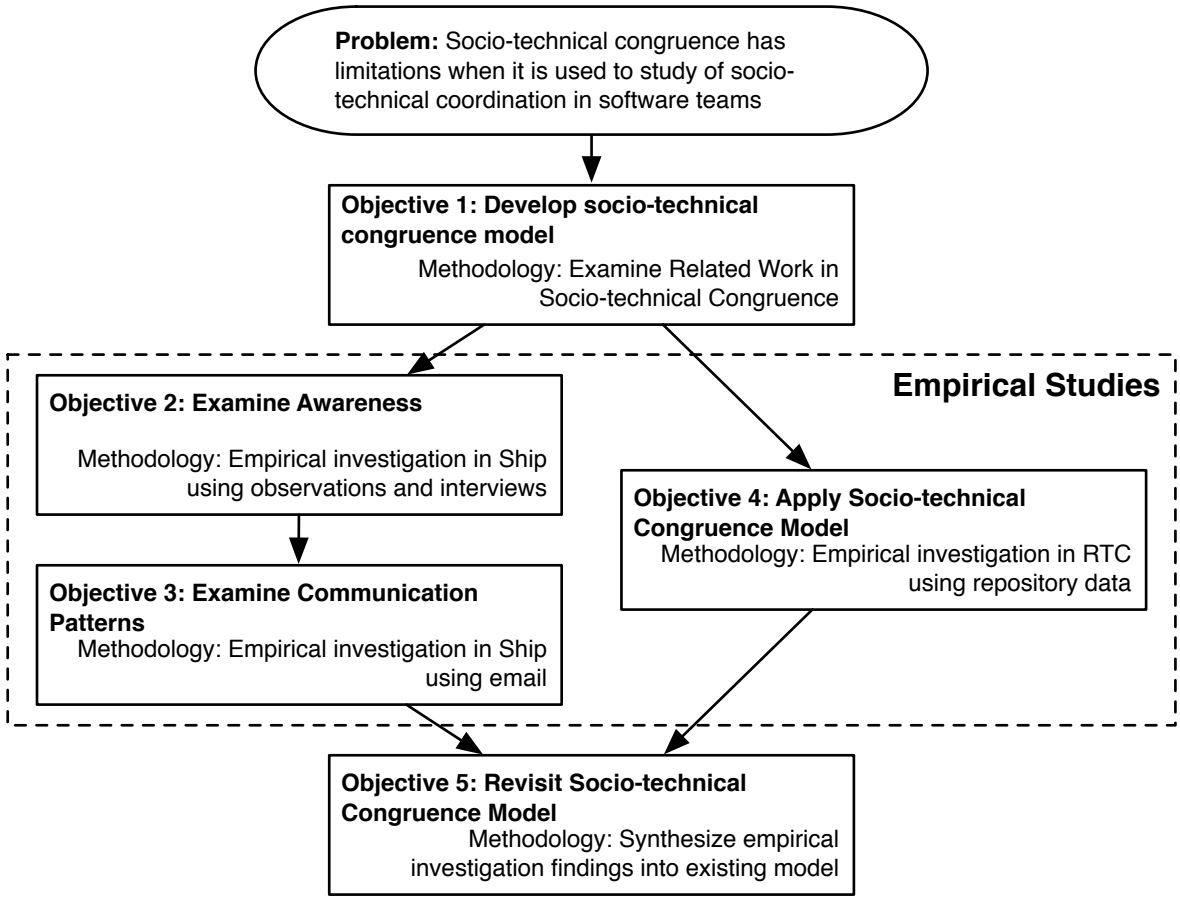


Figure 1.4: Research Methodology

5. I revisited the socio-technical congruence model and refined it according to the findings from the empirical investigations (Chapter 7).

1.5.1 Developing A Model of Socio-technical Congruence

To develop a model to study socio-technical coordination, I used the **socio-technical congruence** measure established by Cataldo et al. [50] as a foundation for the model. By empirically studying social interactions in software-engineering cases, I was able to incorporate insights about coordination into the socio-technical congruence model.

> **Objective 1: To develop a socio-technical congruence model that explains what conceptualizations belong in the model, what elements belong in the model, and how to calculate alignment.**

Based on related work in team cognition and software-engineering coordination, I developed a theoretical model of socio-technical congruence. I also examined existing

studies that use socio-technical congruence. However, the initial model still had limitations in how to incorporate aspects such as awareness, as well as exceptional situations such as important communicators and emergent people.

1.5.2 Empirical Investigations

To address the limitations of the socio-technical congruence model, I conducted empirical investigations on two industrial software teams. The first team, called Ship, was a small global software team in a large corporation. The second team was IBM® Rational Team Concert®[1] (RTC).

I describe below the reasons why these projects were selected for the research. In the following sections, I address how the empirical investigations address limitations of socio-technical congruence.

Project Selection

Ship and RTC were chosen because they were recent global software projects working on complicated systems. The project teams were accustomed to distributed development. RTC has seven sites; the Ship project involved two development sites in the primary team, though others not in the immediate team were distributed as well. Both projects were industrial projects within large, multinational companies. The projects, with respect to both team size and code base, were large enough to make coordination a relatively complicated task. In these respects, both Ship and RTC were suitable projects within which to study socio-technical coordination.

These cases were also selected partly because of convenience. Gaining access to industrial projects is difficult in software engineering research. The opportunity to enter Ship and to closely observe the team members working with each other was an extremely rare opportunity. The head of IT within the organization endorsed the research that I was conducting and gave permission to the employees to meet with me. This buy-in from management is usually very difficult to acquire. For RTC, IBM provided a repository that included a year's worth of issue-tracking data, including comments and build results. Managers at IBM endorsed our work and provided us with the opportunity to speak with the RTC team members.

Rather than examining a single project, I decided that there was more value in investigating multiple projects from different perspectives. Thus, the data collected from RTC and Ship differed in many ways.

Conceptualizing Awareness Using a Socio-technical Congruence Model

To study awareness in a global software-development team, I examined the Ship project *in situ* to gain an understanding of its awareness practices and their use for coordination.

[1]IBM, Rational, Jazz and Rational Team Concert are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

> **Objective 2: To examine awareness and to identify how awareness within a global software-development team can be conceptualized within a socio-technical congruence model.**

Completing this objective provides the information to refine conceptualization of social relationships and addresses the lack of awareness conceptualizations by providing an understanding of how coordination occurs in practice within a software team. Examining these relationships also provides information about what information should be modelled in the socio-technical congruence model.

The research questions for this study are exploratory in nature. They are:

- Research Question 2.1: How do team members maintain awareness of each other's activities in a global software-development team?
- Research Question 2.2: What actions do team members take during an awareness breakdown in a global software-development team?

In this empirical investigation, I reported on how the Ship team maintained awareness, and reported on awareness breakdowns based on direct observations, on interviews, and on qualitative examination of their email archives.

Representing Communication Patterns in Socio-technical Congruence

To study communication patterns in a global software-development team, I acquired email messages from five team members in the Ship team as part of my awareness study.

> **Objective 3: To examine communication patterns, which includes important communicators and emergent people, and to identify how communication patterns can be represented within a socio-technical congruence model.**

Completing this objective provides information to model exceptional situations such as those involving important communicators and emergent people.

The research questions in this study are exploratory in nature. They are:

- Research Question 3.1: What are the characteristics of important communicators within a global software team?
- Research Question 3.2: What situations in email involve of emergent people?

In this empirical investigation, I modelled communication as email, and investigated email communication patterns among team members in Ship. I studied socio-technical coordination by examining the team's work-related email messages. I identified *important communicators* and what their role within the team was using social network analysis, as well as *emergent people* using qualitative methods to form an initial model of their involvement in communication.

Applying Socio-technical Congruence to Study the Relationship With Team Performance

I applied the socio-technical congruence model to investigate the relationship between socio-technical congruence and the performance of a global team working on coordination-intensive activities, and to investigate the applicability of a proposed gap size measurement to investigate a empirical investigation.

> **Objective 4: To apply the socio-technical congruence model to determine the relationship between socio-technical congruence and the performance of a global software-development team working on coordination-intensive activities.**

Completing this objective provides me with knowledge that a preliminary application of the model provides valuable and useful results for coordination, and illustrates the value of gap size and weighted congruence extensions in the socio-technical congruence model.

The research questions are:

- Research Question 4.1: Does socio-technical congruence have an effect on build success probability?
- Research Question 4.2: Does gap size have an effect on build success probability?

I conceptualized a coordination-intensive activity as a software build. A software build either succeeds, meaning that the team was able to perform well and integrate successfully, or it fails, meaning that the team had a coordination problem. Thus, I was able to use traceable connections in the RTC repository to identify if *socio-technical congruence influences the probability that a build is successful*. I also applied a *gap size calculation* that I developed as a part of the socio-technical congruence model.

Previous work in socio-technical congruence [50, 48] established a precedent that increased socio-technical congruence correlates with increased performance within software teams. Thus, in the RTC project team, I hypothesized the following.

> **Hypothesis 4.1: Increasing socio-technical congruence correlates to increased team performance.**

I examined the effects of a gap size and its relationship to team performance in RTC. A gap in socio-technical congruence suggests that there is a lack of coordination where coordination should be occurring, and thus could be considered as a negative influence. Ehrlich et al. [84] found that increased numbers of gaps led to increased code check-in incidents. Following this logic, a large gap size would be considered less desirable than a small gap size or no gap size at all. I hypothesized the following.

Hypothesis 4.2: Decreasing the gap size correlates to increased team performance.

In addition to investigating these hypotheses, I also examined additional influences on build success probability within the context of the RTC project.

1.5.3 Revisiting the Socio-technical Congruence Model

My final research objective is:

Objective 5: To improve the socio-technical congruence model developed in Objective 1 by incorporating the results of the empirical investigations from Objectives 2, 3, and 4.

Using the empirical results of the empirical investigations, I incorporated the findings into the socio-technical congruence model. I refined the existing model's conceptualizations and provided guidelines on how to apply socio-technical congruence to study socio-technical coordination.

Completing this objective fulfills the research goal outlined in this dissertation.

1.6 Contributions

The contributions of this thesis are the socio-technical congruence model and empirical findings from the studies. The findings from the Ship empirical investigation and the RTC empirical investigation revealed a number of empirical findings directly related to socio-technical coordination. These results led to the improvement of a socio-technical congruence model for software-engineering research.

1.6.1 A Model for Applying Socio-technical Congruence

The main contribution of this dissertation is a **model that describes how to apply socio-technical congruence for studying socio-technical coordination that addresses the limitations of existing socio-technical congruence techniques**. This model integrates empirical findings of social coordination that I observed during my studies. The model guides a researcher who is planning to apply socio-technical congruence to examine software teams and performance outcomes that may be affected by social and human factors.

The elements of the model are refined **conceptualizations of social relationships and technical relationships**, **techniques for calculating alignment**, and methods for **modelling exceptional situations** (Figure 1.5). The model contains *guidelines* advising how to select appropriate technical entities and social relationships when studying a global software development team, and a *template* to enable comparison of different studies. It suggests including emergent people by adjusting technical

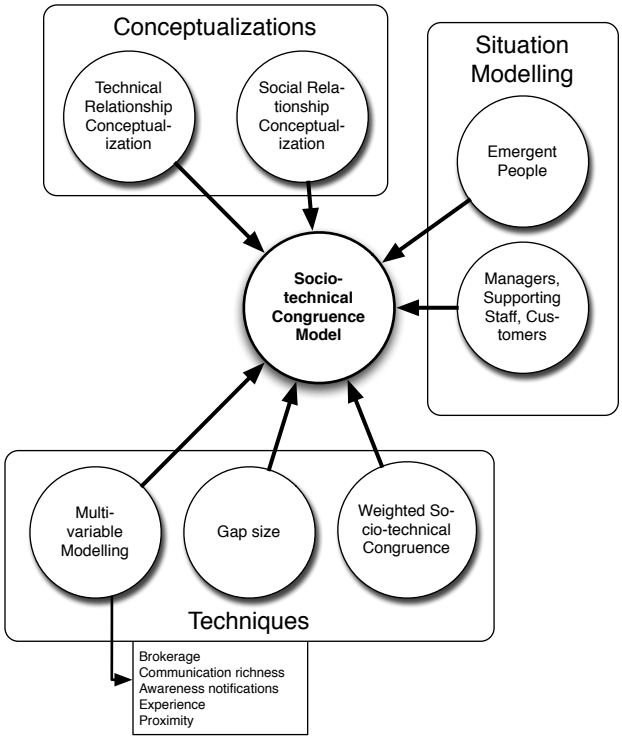


Figure 1.5: Elements of the Socio-technical Congruence Model

relationships over time and including important communicators by broadening technical relationships to include these relevant people. It also provides instructions on using alignment techniques such as weighted socio-technical congruence, gap size, and multi-variable modelling to accurately represent the team and its interactions.

1.6.2 Empirical Findings

The other contribution of this dissertation is a number of empirical findings about socio-technical coordination in software-development teams.

Team members coordinated using multiple forms of media, often simultaneously. Many types of media were used to acquire information, including phone meetings, emails, instant messenger, and personal visits. Developers also used the integrated development environment to study traces left by others on the environment, and were able to work on source-code changes based on this information without necessarily having to engage in communication. Often, multiple media types were used simultaneously by the team members to communicate different types of information.

Experienced team members were aware of the tasks of others such that they could identify and bridge awareness gaps by coordinating. Experienced team members recognized when team members had coordination issues with

other team members, and were able to help bridge these gaps and enable communication.

When awareness breakdowns occurred between team members, experienced team members were swift to recognize these breakdowns and take appropriate action informing others about the situation.

Socio-technical coordination in software-development teams involved expert knowledge sharing, but also planning and routine tasks that involved managers and support staff. Much of the information exchanged in a software-development environment involved expert knowledge, planning and coordinating activities, and requesting or providing permission to execute tasks. Environment coordinators involved with setting up the IT infrastructure of the team were especially important team members.

People were found to emerge into a discussion due to *unexpected events*, *requests from others*, *announcements*, and *following-up on events*. Identifying these situations illustrated why experts appeared in conversations. Similarly, emergent repliers were found to reply to a discussion because they *consulted with experts*, *provided resource or status updates*, *reported results*, or *provided expertise*.

The context of a coordination-intensive activity influenced whether socio-technical congruence is beneficial for that activity. Socio-technical congruence had different effects on certain types of software builds. In particular, socio-technical congruence correlated with the probability of success for continuous builds, but was inversely correlated with the probability of success for integration builds. Integration builds involved a large number of components from different teams, whereas continuous builds involved changes from within a local functional team. The degree of awareness in RTC may have affected how they coordinated their builds. The important result is that increasing socio-technical congruence does not necessarily improve an organization's performance—context and conceptualizations affect performance outcomes.

Communication between individuals occurred when problems arose. Developers were more likely to communicate proportional to the number of dependencies they had with others in failed builds than in successful builds. In other words, there was a larger amount of communication relative to the number of technical dependencies between developers in a failed build compared to the amount of communication in a successful build. This suggests that the developers have an idea of when build will fail, and increase communication along their dependencies to compensate for this failure; alternatively, a complex build that is likely to fail may also have required more communication among developers than a simple build.

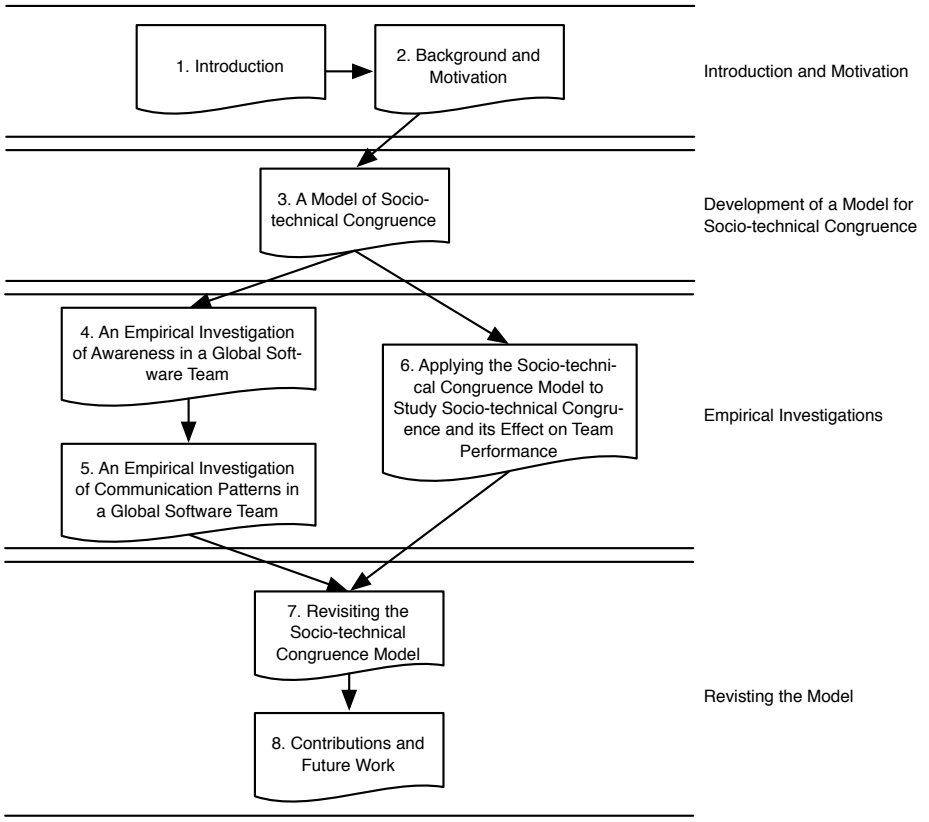


Figure 1.6: Dissertation Chapters Layout

1.7 Dissertation Structure

The remaining chapters of this dissertation are arranged as follows (Figure 1.7).

Chapter 2 outlines the existing work in software-development team coordination, with a particular emphasis on socio-technical coordination and communication.

Chapter 3 describes an initial socio-technical congruence model based on descriptions and empirical investigations of socio-technical congruence from related work. It also outlines the limitations with socio-technical congruence that must be explored further.

Chapter 4 describes a study of awareness by examining the Shipping system project. I describe the information that is exchanged between team members, how they maintain awareness of each other, and instances in which awareness broke down.

Chapter 5 describes a study of communication by examining the Shipping System project. I describe the profiles of important communicators within the team, as well as situations around emergent people who are involved in email discussions.

Chapter 6 describes the Rational Team Concert empirical investigation. In this empirical investigation, I examine the effect of socio-technical congruence on team performance. I apply the socio-technical congruence model and a weighted congruence

measurement.

Chapter 7 presents the revisited socio-technical congruence model inspired by the results of the empirical investigations. This model presents refinements to conceptualizations and guidelines about how to apply socio-technical congruence to study socio-technical coordination.

Chapter 8 concludes the dissertation and discusses future work.

A glossary of terms appears at the end of this dissertation after the appendices for the benefit of the reader.

Chapter 2

Background and Motivation

Current software systems require contributions from tens or hundreds of people who may span multiple offices, cities, or even continents [116, 73, 72, 26]. Reasons that global software development has taken off include acquiring specialized talent from overseas markets, cost-savings, and reducing the distance to the customer [41]. However, global software development is known to increase the amount of time required to solve issues [116], and is subject to a number of social issues, such as cultural considerations [59, 167, 122]. To build such systems, we need to ensure that the team is not only capable of developing components of a system, but also has the governance to be able to integrate the interdependent parts into a whole. As a consequence, it is important to be able to understand how people coordinate their work in such an environment.

In this chapter, I explain the importance of studying socio-technical coordination.

2.1 The Importance of Coordination

Software engineering is largely a design discipline [64, 115] and is extremely labour-intensive [32, 51]. As a consequence, the average worker in software engineering usually ends up being an expert in some element of the system. Effective coordination and agreement among these team members thus becomes a prerequisite to effective software development [133].

Coordination is a well-established concept with a consistent set of definitions. Van de Ven defines coordination as "integrating or linking together different parts of an organization to accomplish a collective set of tasks" [213], which takes a top-level organizational view of the issue. Malone and Crowston define it as "managing dependencies between activities" [146], which makes explicit the concept that if there are no dependencies, then no need for coordination exists.

Research in software-engineering coordination has examined interactions among software developers [42, 151], how they acquire knowledge [85, 163], and how they cope with issues such as geographical separation [89, 111]. The ability to coordinate has been shown as an influential factor in customer satisfaction [133] and improves the

capability to produce quality work [91]. A study by Nagappan et al. [162] showed that elements of the organizational structure were able to failure proneness in a large product. Similarly, communication patterns have also been used to predict build quality, suggesting that communication structures have an influence on build outcomes [223].

One of the objectives of software-engineering research is to improve the performance of software-development teams. By making these teams work together efficiently, software can be delivered in a timely manner, with fewer defects and less waste. Coordination is an influencing factor on software project outcomes [90, 64] and on the performance of software teams [133, 89], but can be difficult to conceptualize and measure.

2.1.1 Coordination in Global Software Development

The problem in global software development is that communication over distance is difficult. Response time of remote teams is delayed [116, 69] and cultural and social issues interfere with a team's communication. The awareness that team members have of others is limited and thus are reliant on explicit communication [103] or communication through bug-tracking and code workspaces [186, 104, 20]. Time zone issues affect product quality [43] as well as global organizational structuring [156]. Cultural factors may also come into play, affecting the quality of the team's cognitive understanding of the issue at hand, or their interaction with each other [113, 167, 122].

There is evidence that a unified team using an effective tool can mitigate some of the effects of distance by providing awareness [166, 20]. Another study observes that failures are not affected by distance [28]. Some of the factors that are discussed as affecting the effects of distance include a symmetric relationship between the sites, synchronous communication, consistent tool use, good organizational integration, and strong feelings of ownership for the end product [28]. Note that these factors indicate that the team has a good understanding of the task, the team, and the interactions, and thus indicate that the team shared an effective mental model.

Although distance has been identified as a challenge, advances in collaborative development environments are enabling people to overcome challenges of distance. Technology that empowers distributed collaboration include topic recommendations [42] and instant messaging [168]. Processes are adapting to the fast pace of software development: the Eclipse way [95] emphasizes placing milestones at fixed intervals and community involvement. One study of early RTC development shows that the task completion time is not as strongly affected by distance as in previous studies [166]. Bird et al. found that distance between teams did not significantly affect the number of end-user failures in binaries [26]. Existing research reports that team efficacy leads to improved performance [153], and literature in distributed software engineering speaks frequently about the benefits of building trust among the teams [41, 119, 100, 153]. Though distance has been shown to change the way that people need to coordinate, the impression is that distance is the new reality, and development teams are coping with the changes.

2.1.2 Studying Socio-technical Coordination

The phrase "socio-technical coordination" was used by Herbsleb [112] to describe key issues regarding software engineering research. He states that global software development is the future of socio-technical coordination, and that the "key phenomenon of global software development is coordination over distance".

An ongoing need is the ability to examine team interaction and coordination in a global environment. Because communication across distance can be expensive[116], and awareness overload can lead to problems [69], ensuring that coordination is not only effective, but efficient is important. Identifying the relationship between coordination and the performance of a team continues to be a key motivation for research, as well as for the software industry. Thus, being able to conceptualize and measure socio-technical coordination and relate it to performance is important both to researchers as well as practitioners.

2.2 What is Socio-technical Coordination?

Before I further investigate persistent issues in global software development, I investigate the history of "socio-technical coordination". There appears to be no definition in the literature of what precisely socio-technical coordination is. Herbsleb leaves the phrase's definition implicit [112]. In "Global Software Engineering: The Future of Socio-technical Coordination" he describes "technical coordination in geographically distributed projects" and defines coordination using Malone and Crowston's definition—"managing dependencies among tasks" [146]. "Socio-technical" fits the profile of a software-development project, which of course involves social aspects as well as technical ones. One of the points mentioned is the notion of "fit" of a software architecture with the organization that will build the system. This fit refers to Cataldo's work on socio-technical congruence [50, 48], which emphasizes the alignment between the dependencies of the technology that the organization is building, and the organization's communication.

The importance of defining socio-technical coordination is to ensure that researchers have a common frame of reference when it comes to addressing coordination issues in global software development. Many of the concepts overlap heavily in multiple frameworks, especially from the computer-supported co-operative work area. Neale et al. [164] present a model for evaluating "activity awareness" of others, building a base on contextual factors and growing common ground from both coordination and communication. The 3C model [96] defines three categories, "coordination", "co-operation", and "communication" for groupware classification, enabling application developers to classify their work accordingly. Sarma et al. [190] also presents a classification framework for tools based on Maslow's hierarchy of needs. Each of these models and frameworks identify characteristics of existing tools and methods, and whether they address concepts like coordination, communication, and awareness. However, these frameworks do not explain how to conceptualize and study

socio-technical coordination in such environments.

In order to understand socio-technical coordination and the need for such coordination in global software development, I begin by investigating the earliest reference I could find to "socio-technical", which is socio-technical systems [55]. I then examine how coordination is carried out in socio-technical environments, and extend this coordination to related work in distributed socio-technical environments that are related to software engineering. The purpose of this discussion is to ground our understanding of the term "socio-technical system" firmly into the domain of global software development.

2.2.1 Socio-technical Systems Theory

The phrase "socio-technical" appears to come from a theory proposed in the 1970s regarding the interaction between social systems and technical systems. This theory is called "socio-technical systems theory" [36].[1]

The term "socio-technical system theory" was first brought up in the context of management information systems (MIS) [55]. Socio-technical systems, alternately known as socio-technical theory, is a theory in organizational research regarding the interaction between humans and technology in the field of business efficiency and productivity. Bostrom and Heinen describe the socio-technical systems approach as follows [36]:

> A work system is made up of two *jointly* independent, but correlative *interacting* systems—the social and the technical. The technical system is concerned with the processes, tasks, and technology needed to transform inputs to outputs. The social system is concerned with the attributes of people, the relationships among people, reward systems, and authority structures.

Bostrom and Heinen continue to state that [36]:

> The STS approach argues that any design/redesign of a work system must deal jointly with the social and technical systems.

MIS, in particular, has a direct influence on the technical system, which contains the technology and the tasks, or processes, within in the organization (Figure 2.1). The reason that this is interesting is because any software-based tools that are introduced will affect technology and tasks. When we extend the organization to a global software organization, the influence of technology, which is influenced by MIS, grows immensely.

[1] Interestingly, "supporting congruence" is a principle that is mentioned in one of the early references to socio-technical systems theory [55], though in their context it refers to the idea that the system must be designed so that it matches the values of the organization. This definition of "congruence" is somewhat different from the formal "socio-technical congruence" defined by Cataldo et al. [48].

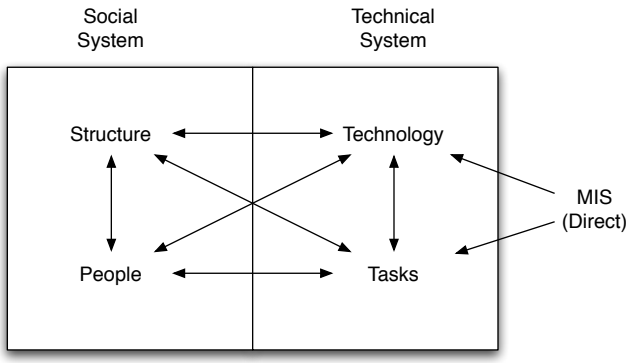


Figure 2.1: Interacting Variable Classes Within a Work System from Bostram et al. [36]

Bostrom and Heinen [36] describe a number of conditions that are the major causes of inadequate designs and unsuccessful change strategies within the field of MIS, and argue these conditions as the rationale for why such an approach that integrates both the social systems and technical systems is essential. These conditions include such points as the "implicit" theories held by systems designers about organizations, the weak conceptualizations for organizational work systems, the limited view of the goal of an MIS implementation, the inability of the designer to include relevant people when examining the system's needs, the static view of the systems development process, and the limited ability for a designer to engage organizational change.

In the past thirty years, a number of these issues have been remedied to some extent, or if not remedied, at least investigated extensively in software engineering and management information systems research. For instance, the requirements engineering discipline has largely identified the importance of involving the end user, and agile software processes have made systems development more flexible.

Of course, the large, mainframe-based MIS systems of the 1970s and 1980s are not quite the same today. Gone are large server rooms—now, a desktop exists on each person's desk. Working at a computer with network access is considered mandatory equipment in a software development company. Yet, despite the shift in MIS from large, monolithic systems to systems that are focused on individuals, the variables within the organization still hold relevance. Perhaps even more relevant is that the designers of MIS no longer design for their own organization only, and in this respect, considering these variables' effect on the organization becomes even more poignant. More than ever, the interactions between people and technology affects the business.

Currently, the term "socio-technical system" generally describes the environment in which human beings work (e.g. [169]). "Socio-technical" has been occasionally applied to software engineering; one such case being in Al-Rawas and Easterbrook's description of communication problems in requirements engineering [14]. Again, the phrase is not defined or attributed to a source.

Thus, an appropriate definition of a socio-technical system is as follows. *"A socio-*

technical system is an organization or a sub-organization that requires interaction between humans and technology to carry out its assigned goal." The key elements in the definition is that humans and technology must interface, and that the people working together generally function as a team.

The implication of the socio-technical theory is that a manager should remember that the technology and its interaction with the environment has an impact on the organization, and that a systems designer should consider human aspects, such as the user experience, when designing systems. Failing to do so results in inadequate software systems that fail to meet the needs of an organization. Though this problem was identified was early as 1976, it remains relevant today.

2.3 Teamwork in Socio-technical Environments

The formalization of socio-technical theory elevated the human to a top-level entity, and focused particularly on the importance of a manager or systems designer in affecting the system. As a single person's knowledge of a system is specialized as well as limited, that person often is unable to make the appropriate modifications in dependent components when a component is changed. Coordination is necessary to ensure that a client's requirements are being met [14, 73] and software team members must coordinate within teams as well as among teams to ensure that each component integrates successfully into the system[175, 115, 113, 78].

Many models and theories have been proposed to describe the forces that allow a team to work together. The concept of some kind of group-based knowledge is well-researched (e.g. [131, 53, 57]). Understanding the cognition that occurs within a team provides some groundwork for understanding team coordination.

A **team** consists of two or more people who have shared objectives and are willing to work to accomplish a common goal [184]. In addition, Cannon-Bowers [39] distinguishes a team as having clearly-defined roles and task-specific knowledge. I specify the definition of a "team" for the same reasons as Cannon-Bowers [39] and Cooke [57] do: A team is different from a group because the team has a common goal, and each member of a team has a distinct set of role and specialized knowledge, whereas a group usually refers to a large number of people. Implicitly, the team members also have dependence on each other—this dependence implies the need for coordination within the team.

Much work has been done on cognition of teams in the workplace, especially in the naval and military contexts [220, 123, 109, 169, 197] but cognition has also been examined within design teams [39, 161, 21]. These studies each examine how a team interacts with each other and their environment in order to accomplish a design-oriented set of tasks. Various labels and models have been applied, such as distributed cognition [124], team knowledge [57], and team mental model [131]. However, all of these labels generally refer to the idea that aspects of cognition within a team, especially in relation to performance, differ from individual cognition [153].

The "mental model" [131] in particular has received attention as an explanatory model referring to one's understanding of complicated schemes. The mental model was adapted to the team setting by Cannon-Bowers et al. [39]. In the team context, a mental model is a knowledge or a belief structure. There are two models that a team develops: a task model, which represents the team's understanding of the task; and a team model, which represents the team's understanding of the capabilities of the team members. Klimoski and Mohammed [131] detail the meaning of a "team mental model" as "organized mental representations of the key elements within a team's relevant environment that are shared across team members".

Cannon-Bowers et al. go into more depth and proposes a taxonomy of four mental models: the task model, the equipment model, the team model, and the team interaction model [39]. This model is well-suited in a socio-technical system, as it separates equipment into a separate model. Interestingly, the elements that appear in this taxonomy reflect quite closely the interacting variables described by Bostram et al. [36] (Figure 2.1).

Some of the key points of studies of communication from a cognitive point of view allow us to identify what interactions occur between individuals, as well as between individuals and their technical environment. Key elements include the propagation of the representative state of the system through speech channels and using artifacts for memory [124] and shared situation rooms [203] that allow team members to form a common understanding of the current contexts.

This informal "creation" of understanding is very important especially with respect to how people interact in shared spaces. For instance, Suchman describes a "situation room" where people work, and discuss explicit exchanges of information, such as conversation, as well as implicit exchanges such as eye gaze [203]. Such a setup allows for the people at work to engage in what is referred to as a "common focussed gathering" [sic], in which everyone's attention is directed toward one particular scene. These implicit cues are not limited to socio-technical environments; they have been used as the basis for the design of robots [197] that interact with humans.

In this dissertation, I will use the term *team mental model* as a general encompassing phrase that refers to a person's mental model of the team and team interactions. There is a body of knowledge focused on shared mental models on design teams [39, 21], on team performance [161, 153], and on collaboration [39, 87] that each reflect on the team mental model.

2.3.1 Benefits of a Team Mental Model

A literature review of the team mental model construct concludes that having a unified team mental model benefits team performance [161]. Other reviews and studies (e.g. [153, 130]) support this statement. One of the ways in which the mental model accomplishes this is because the team engages in reinforcing behaviour. For example, Oser et al. [172] found that offering information before it was requested promoted team effectiveness, and previous work has shown that mental models can

provide a set of organized expectations from which accurate, timely predictions can be drawn [202, 182]. Cannon-Bowers et al. describe one of the primary benefits of a shared mental model as being the creation of not necessarily the model itself, but of "shared expectations" [39].

Common training programs that teach team members how to coordinate efficiently, without coordination overhead, is one way to achieve a team mental model. Team members with training were able to better anticipate each other's information needs [87, 202].

2.3.2 Coordination and the Team Mental Model

The team model and the team interaction model refer to an understanding of a team member's abilities, as well as the communication patterns that occur within the team. Espinosa et al. made the link between shared mental models and coordination in software development explicit [88, 89], describing a number of propositions, including "shared knowledge of the task and shared knowledge of the team help team coordination".

Coordination can be executed in one of two ways, according to March and Simon [147]: by programming or by feedback. **Coordination by programming** in this case means that every participant follows a prescribed process. **Coordination by feedback** involves people adjusting their actions based on receiving new information.

The terms that March and Simon use are somewhat overloaded in socio-technical environments. For instance, "programming" in the organizational sense implies process, but part of a process may involve reacting to feedback.[2] In a socio-technical environment, reading a number from a digital display could be considered feedback, but part of the programming may be to react in a certain way depending on what the digital display says.

Thus, I choose to substitute the word "programming" with **protocol**, and "feedback" with **communication**, and clarify their meanings below.

Protocol

Protocol is executing actions according to a pre-defined set of rules. For example, a number of cars stopping at a controlled interaction run by traffic lights do not communicate per-say, but they react to the traffic lights. The rules that "red means stop" and "green means go" are protocol rules that are widely understood.

In many organizations, many protocols are not explicitly defined, and in fact the members may not consciously recognize what they are doing as protocol. This has been illustrated by the studies cited above [124]. This is where the concept of "team mental model" comes in: the more unified the mental models are of the team members, then the more likely that they are able to recognize and follow the protocols.

[2] In addition, the word "programming" in software development means, "writing a program"!

Protocol exists in most software engineering groups. One of the most commonly-used protocols is source-code version control, such as CVS [3], SVN [6], git [8], and Mercurial [11]. The team requires that the code written at a developer's individual workstation be checked into some kind of repository, and that others will retrieve the code once it has been checked in.

Protocols may also happen at a higher level, such as through project planning and issue-tracking (e.g. JIRA [7], Bugzilla [9]). Many are implicit; even agile teams, which make a point of avoiding heavy processes, have protocols regarding feature release planning and coordination support [198].

Communication

Communication is the act of sending information to another person. Communication is explicit, intentional, and involves at least two people. Talking to someone face-to-face, sending someone an email message or writing a comment directed to them in a repository are each considered intentional and direct communication, which I call *explicit communication*. Passive methods of communication, like being present in the office and therefore leaving evidence of your presence to others, is *implicit communication*. Other examples of a person engaging in implicit communication include that person reading a field manual whose author is not known or that person making an action in a repository that results in the software system sending out an automatic notification to others broadcasting the action.

2.3.3 Awareness as a Reflection of the Team Mental Model

The social cues and hints that a person receives from the environment and from other team members contributes to their ability to coordinate. This concept is captured in our earlier descriptions of the team mental model. However, the team mental model refers to a person's internalized, long-term knowledge of how a team works and does not describe a person's immediate knowledge of a situation.

Dourish and Bellotti define awareness as "having an understanding of the activities of others .... that form a context for your own activities" [80]. Awareness has been described as a type of knowledge. Espinosa distinguishes between awareness as "fleeting knowledge", which is the short-term knowledge that an individual has of a situation, and explicit knowledge, which is tacit [89]. Whereas explicit knowledge is often transferred through explicit communication, such as in written documents or email [218, 19], fleeting knowledge can be transmitted through the above methods, but also through less obvious cues such as eye gaze or even online presence indicators [124, 37, 97].

The definition of awareness resembles the definition for a team mental model, which is an "organized mental representations of the key elements within a team's relevant environment that are shared across team members" [131]. Similarly, Cannon-Bowers et al. describe a category in the shared mental model conceptualization as "the knowledge of the team member's knowledge, skills, and abilities" [39]. In these

respects, awareness relates highly to a team mental model. A team mental model is a richer description that describes what a person understands about his team. The distinction is one of temporality: that is, whereas a team mental model is an overall, long-term understanding of how the team works together [131], awareness is knowledge of the current circumstances and situation [89].

Coordination, team mental models, and awareness are interrelated. Entin et al. [87] proposed that ongoing coordination would help the team acquire a shared mental model, and observed that a team with a shared mental model was able to coordinate implicitly. Similarly, Khan et al. [130] found that implicit coordination improves *sharedness accuracy*, that is, a team's understanding of a situation. Heath [109] reinforces that there is no point in time at which awareness is achieved, and that rather it is "ongoingly achieved in collaboration with others". Espinosa et al. [88] conducted a study of awareness and conclude that shared mental models contribute to software development, and that familiarity with technical artifacts contributed to the shared mental models of both the task and the team. In another study with a similar setting, the same authors found that shared knowledge of the task, as well as task awareness and presence awareness helped aid coordination in software development [89].

The use of electronic communication media, especially written email, severely limits social context cues [217]. Yet, at the same time, the ease of sending electronic messages can enable individuals to become more aware. Fussel et al. [97] postulate that having awareness of others' activities, for example by being CCed on email, helps others coordinate their work. Garton and Wellman [99] identified that those who use email are more aware of events in an organization than those who do not use email. The phenomenon of "information spillover" in email is mentioned by Kraut and Attewell [134] as one of the reasons why those who extensively used email were more informed in an organization than those who did not.

Social presence theory [217] refers to the feeling that other actors are involved in the communication. In an interpersonal context, this phenomenon has been labelled **presence awareness** [37]. These clues are often provided through nonverbal cues, such as physical presence, open or closed doors, and lights being on or off. There have been numerous attempts to introduce clues that would ordinarily be available in proximity to online tools.

Awareness in Distributed Software Development

Due to issues with communication across large distances, a lack of awareness in software development can lead to rework and confusion within the team [69]. If someone is not aware of changes made to a requirement, then that person may not proceed on working on a task, or worse, may overwrite previously-made changes. If critical information is not processed in a timely fashion, the coordination breaks down and mistakes in development can occur [138]. Awareness problems are especially pronounced in global software development, where distance reduces awareness clues and increases the reliance on digital communication [72].

A study of a distributed team found that distance affects the perception of awareness across team boundaries [69]. Similarly, Cataldo et al. [46] conducted a study on "coordination breakdowns" in a distributed team, and identified a number of situations where team members failed to communicate with each other, resulting in integration problems. Their study identified that the dependencies between different modules, and the lack of coordination around them, were prime causes of technical issues within the project. However, Erhlich and Chang [85] examined different facets of awareness, concluding that developers who were familiar with each other in the past were more likely to consult with each other, even across project boundaries and distance.

Developers become aware of the system and of others' activities through mechanisms such as APIs [78] and through actions they leave on shared artifacts [103, 34]. Issue-tracking tools [102, 54, 166] are particularly common for task awareness. Awareness within source-code control systems have been studied as well though there is not much evidence of its widespread adoption [186, 189].

2.3.4 Communication in Distributed Software Development

Software developers spend much of their time communicating [175]. Because developers face problems when integrating different components from heterogeneous environments, developers engage in direct or indirect communication, either to coordinate their activities, or to acquire knowledge of a particular aspect of the software [163]. Herbsleb et al. examined the influence of coordination on integrating software modules through interviews [113], and found that processes, as well as the willingness to communicate directly, helped teams integrate software. De Souza et al. [76] found that implicit communication is important to avoid collaboration breakdowns and delays. Ko et al. [132] found that developers were identified as the main source of knowledge about code issues. Wolf et al. [223] used properties of social networks to predict the outcome of integrating the software parts within teams.

Meetings and phone calls are common, as well as informal "water cooler" discussions [133]. However, this kind of interaction cannot occur over distance. Telephone conferencing and video conferencing displays help mitigate this by providing synchronous communication similar to face-to-face discussions [81, 40, 71].

Across distributed offices, the team members may use formal documents such as project plans and written documentation [176, 143] or perhaps Wikis [67]. These written communications are extremely important to facilitate communication with distributed sites [156]. They may use less-formal methods of communication including issue-tracking systems [102, 103, 166, 10, 5], email [158, 103, 206], instant messenger [106, 125, 168], real-time displays [42], or even microblogging [193].

Email Communication within Socio-technical Organizations

Email is prominent in modern society. In AOL's Fourth Annual "Email Addiction Survey", 46% of email users said they're hooked on email, 51% check email four or

more times a day, and 62% of people check work email on the weekends [4]. As early as 1988, Sumner [204] observed that email is useful for providing information updates and scheduling activities, and that with respect to managerial functions, 41% of sent messages and 33% of received messages were "organizing", which consists of requesting information, notifying others of decisions, and replying to requests. Dabbish and Kraut report that email is generally viewed as very important in projects that have high task interdependencies [65]. Employees at an organization were interrupted almost every five minutes by email, and responded within six seconds [127].

Many software-development organizations use email for communication. Email is considered a standard method of communication in a number of organizations (e.g. [79, 154, 206, 214]). While I know of no specific study that surveys the use of email in software-engineering teams, independently-executed studies of individual software teams indicate that email is used frequently in such teams (e.g. [119, 206, 74, 18, 154, 142]). Usage varies depending on the company, as well as the geographical location of the office [206]. Open-source projects utilize email heavily, especially through mailing lists [158, 103, 181]. Some empirical studies report on the use of email by employees [154] but in many cases I believe that the wide-scale adoption of email is taken for granted.

Most of the existing research on email use in organizations can be classified into one of three categories. The first category investigates how individuals use and manage their email, with particular attention paid to the tasks that an email system is used for other than communicating with others. The second category are quantitative investigations of email use, with particular attention given to reply prediction. The third category is a more general view of how email affects the organization in which it is used.

There is a body of knowledge that has conceptualized an individual's email use, especially with respect to use of email client software. Research has investigated how an individual organizes messages and uses email systems, [221, 92, 22] and how that person conceptualizes the messages arriving in the inbox [74, 92, 206]. Dabbish et al. [66] identified that individuals use email for four purposes: project management, task delegation, and reminders; information exchange, storage, and retrieval; scheduling and planning; and social communication. This style of research directly led to proposals of improved tools and visualizations for managing email. [23, 214, 215, 93]. There are very few distinctions on how certain organizational roles use email, other than the observation that managers tend to use email in a slightly different manner than others in the organization.

The next level of detail looks at how people use email to communicate with others. Reply analysis and prediction is a quantitative way to examine email and provides some insight into communication patterns in an organization. Dabbish et al. [66] identified a number of factors that appear to influence whether a reply is sent, including the communication frequency with the sender. A large number of recipients actually reduced the probability that the message would be replied to. Karagiannis and Vojnović [128] approached email using a quantitative perspective and provided a

technique to predict response times of email; some of their findings included the revelation that only 15% of users reply to more than 10% of their email. Their findings also corroborated Dabbish's earlier result showing that an increase in the number of recipients for a message decreases the probability of a reply.

Research also reports on the perceptions of email by users. Tyler et al. [210] examined behaviour of email users especially in the context of interacting with others and found that users were sensitive to issues such as their availability. For example, a user will reply in a particular manner to set expectations on their receivers. Tyler also reports on the use of multiple communication methods that augment email; for instance, using email and voice mail in conjunction to send information to a receiver. Dabbish and Kraut found that the feeling of being overloaded with email could not be predicted by the number of messages in their inbox [65]. The same study found that email volume was greater if the person belonged to a higher number of projects or if their management duties were greater.

In general, email is an equalizer. In a survey of research findings regarding email, Garton and Wellman state that it tends to allow members of a group to communicate more easily, links peripheral workers with the core, and makes remote people more easily accessible [99]. Email also allows an organization to have a more fluid team structure, and allows access to employees and knowledge who otherwise may be inaccessible [99, 134]. Aranda and Venolia [18] into coordination patterns between developers has also provided insight into email usage. Based on their observations and interviews with developers, they observed eight coordination patterns that are related to email, including "shotgun emails" where someone sends messages to a large number of people hoping someone will reply and "snowballing threads" which involve adding people to an increasing list of recipients.

There is not much related work on email patterns in software engineering companies. Related work has concentrated primarily on email usage by individuals [222, 177, 66], or on mechanistic aspects such as inter-reply times [210, 206, 128]. Aranda and Venolia have gone toward characterizing some patterns in mailing lists [18]. Most of the work on email examination in software engineering uses social network analysis. Social network analysis allows one to examine specific individuals in relation to their neighbours. The characteristics of these individuals may allow one to learn about their interactions, and in turn may influence the socio-technical congruence model conceptualizations regarding important communicators within a network.

Social Network Analysis

Recently, there has been interest in social structures in software engineering teams using social network analysis. Tyler et al. suggested a technique based on betweenness centrality to identify communities of practice [208], and was able to identify that leaders were more likely to end up in the centre of a communication network. Rowe et al. also use community structure and interconnections to determine the organizational structure given email information [183]. Communication structures have been applied

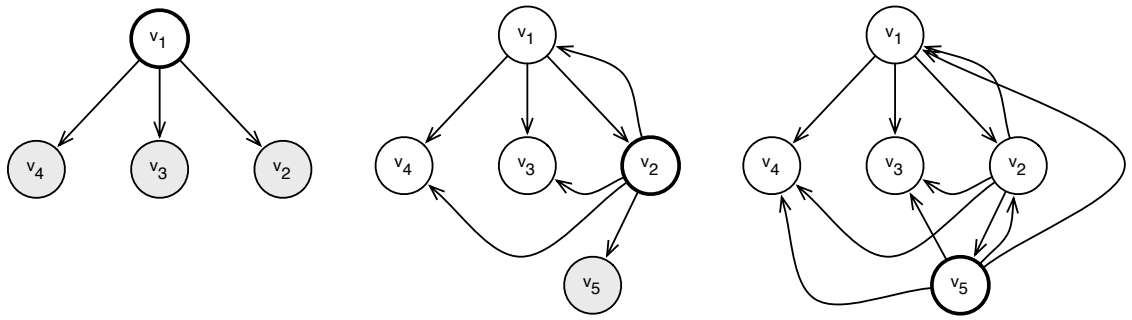
to various organizations to discover a number of interesting properties, such as how brokers between teams are known to be purveyors of good ideas [108, 38]. Hossain et al. [120] also identified that highly-centralized actors tend to be more involved in coordination utterances than others. Yang and Tang [225] demonstrate that poorly-performing teams tended to have "unbalanced" communication structures. Hinds and McGrath [118] identified that team members in loose structures are better able to coordinate work.

Social network analysis offers a method of analysis beyond simple generalization of group behaviour, focusing particularly on the differences between individuals [33]. Because an email message originates at one person and is sent to one or more recipients, social network analysis is appropriate for understanding communication patterns in email. A social network is a structure that represents relationships between people. A social network can be represented as a graph G such that $G = V, E$ where V is a set of vertices or actor and E is a set of edge which are two-element subsets of V.

The Enron data set is a popular target for analysis as it is corporate data that is now publicly available, and many analyzes using this data source have been performed (e.g. [52, 129, 60]. The most relevant is Diesner et al. [79], who observe the behaviour of a company in crisis using dynamic social network analysis, and conclude that the communication patterns become much busier during a crisis affecting the company. Creamer et al. invent a metric called "social score" in the data set to identify "important" individuals. This metric is based on data such as the average reply time, involvement in cliques, and degree of centrality measurements.

Rice et al. [180] report on email adoption within a company using dyadic analysis. By analyzing pairs of people, the authors were able to identify whether both participants tended to adopt electronic messaging or not. In their empirical investigation, they concluded that two people who communicated frequently were both likely to adopt, or not adopt electronic messaging.

Social network analysis has been applied to software development projects, usually to open source mailing lists. Bird et al. examined the participants in an open-source email list [27] and identified key participants based on their network position. Howison et al. [121] applied centrality over time in open-source projects and found that most issue-tracking communications were primarily decentralized. Bird et al. confirmed this effect by studying the existence of a community structure in open-source communication [30], discovering that email communication reflected small working groups of developers. Wolf et al. [223] as well as Bird et al. [29] used socio-technical networks for quality prediction. Crowston and Howison examined open-source software project structures but do not discuss individual roles in the projects [62]. Though some research has investigated communication among roles and particularly in identifying new types of roles based on their small-group interactions [195, 24, 82], there are few investigations about the communication patterns of software-engineering team members. I know of no studies that examine how software-engineering roles, like developers and testers, are involved in communication within a software team, particularly when using email, and what their relationship to the communication structures are.



(a) Sociogram After the Initial Sender v_1 Sends a Message. The Initial Recipients are v_2, v_3, and v_4.

(b) Sociogram After an Emergent Person v_5 is involved in the thread via v_2. v_2 Replies to everyone in the thread, and CCs the message to v_5.

(c) Sociogram After Emergent Person v_5 Replies to Everyone in the Thread. v_5 is an Emergent replier.

Figure 2.2: Illustration of an Emergent People's Involvement in an Email Thread

Emergent People

An unexplored aspect in software-engineering communication is the involvement of people who are not initially involved in interactions [72]. These individuals, for whatever reason, were not deemed to be relevant or were forgotten at the start of the conversation, and are included as recipients at a later time.

An **emergent person** is a person who is involved in an email message thread after the first message is sent. An email thread is an aggregation of email messages with the same subject title. Generally, an email thread contains messages that are replies to a previous message. The initial message is sent by one person to a number of recipients (Figure 2.2(a)). An emergent person is included in the message thread when someone in the thread carbon copies (CC) or forwards (FW) the message to another recipient.

Emergent people may indicate the presence of knowledge that is required for completing the task [69, 72]. An emergent person is interesting in the context of software development because it indicates that, during the course of the conversation, someone who was not deemed relevant to the conversation initially is now important. The presence of an emergent person may suggest that an expert was not available or known to the sender, or it may indicate the involvement of individuals unknown to the team, or unrelated to the task. Emergent people has implications on communication practices in an organization, as well as for expertise-recommender systems, as they may help define the situations in which such recommender systems may be used.

To involve an emergent person in an email thread, an existing participant in the email thread must be added the To or CC header fields. A person may also be involved if they receive a message in the thread as a forwarded message. This situation is illustrated in Figure 2.2(b). An emergent person who replies to one of

the messages in the thread is called an emergent replier (Figure 2.2(c)).

- **Initial sender** The person who sends the first message of a thread. Usually this person sends a message to inform others, or to request information (Figure 2.2(a)).
- **Initial recipients** The initial recipients are the people who receive the first message of the thread. These recipients are explicitly added by the initial sender (Figure 2.2(a)).
- **Emergent person** A person who receives a message in the thread without receiving the first message in the thread. These people are added to the recipients list because a sender either added them to the To field, the CC field, or forwarded them the message (Figure 2.2(b)).
- **Emergent replier** An emergent person who also posts a message in the message thread (Figure 2.2(c)).

In global software-development teams, team members have difficulty identifying experts and seeking knowledge from remote sites [111]. Developers make up for this by regularly communicating [89], but this leads to a tendency to broadcast email messages, causing information overload and leading to communication breakdowns [69]. Damian et al. [72] observed that people assigned to work on dependent requirements often talked not only with people with whom they had technical relationships with, but also with people who had no apparent technical connection to the requirements. Developers acquire awareness of others [89] and frequently seek expertise from colleagues even if they belong to other projects or teams [85]. Consequently, it is important to include the right people in an initial email. In our previous work [69], we observed that team members emerged as knowledge sources during the course of an email discussion—these people were not initially contacted, and were included in the email discussion only after a number of initial messages had been sent.

Seeking information by relying on emergent people may be inefficient, especially if knowledge is not necessarily centred in individuals. Emergent people may also be a knowledge source that can be leveraged. The fact that conversations evolve over time in software engineering has not been extensively studied at this time. Knowing who emergent people are and what they talk about leads us to better understand the process of expert knowledge seeking.

2.4 Research Statement

The problem with studying socio-technical coordination is that there are a large number of social and technical factors to consider. As demonstrated, situations such as varying usage of communication media, configurations of awareness, and emergent people form unique contexts in each organization.

Socio-technical congruence calculates technical relationships derived from technical dependencies among artifacts as well as social relationships in a software team. The idea that coordination should align with the technical relationships of team members is compelling, especially since it reflects the sensibility of Conway's Law [56] and reflects aspects of socio-technical systems theory [55]. The socio-technical congruence framework by Cataldo provides some examples on how to identify work dependencies [44] but does not provide guidelines describing which technical relationships or social relationships to use in order to reflect a software organization, nor does it offer guidelines for interpreting exceptional situations, such as when emergent people appear. Thus, a model should provide guidelines about conceptualizations of social and technical relationships that are appropriate to the organization. The model should also be able to represent any number of exceptional situations, including the nuances of how a team maintains awareness [70], the involvement of important communicators [72, 152], the involvement of brokers [151], and emergent people [72, 69].

There is also a need for more studies of the relationship between socio-technical congruence and the performance of a global software-development team. Existing studies have suggested that this relationship leads to improved outcomes [50, 48, 84]. In addition to this need for empirical studies, there are some conceptual limitations of socio-technical congruence that must be explored, such as the inability for the existing measurement to model gaps. If high socio-technical congruence can be identified as consistently correlating with high outcomes, then it can be used as a concrete recommendation for practice in addition to merely being a tool with which to measure socio-technical coordination.

The goal of this research is to develop the following three aspects within socio-technical congruence for the study of socio-technical coordination in global software teams. First, I examine how I can refine the conceptualizations of social and organizational relationships. Second, I propose and apply improved techniques to calculate alignment. Third, I describe an approach describing how to apply socio-technical congruence.

In the following chapter, I form a preliminary socio-technical congruence model based on related work in socio-technical congruence to in order to better study socio-technical coordination. However, the model will not be able to address all of the concerns stated above regarding congruence, particularly awareness and communication patterns. In addition, the model will also have to be applied to demonstrate its usefulness. Exploring how to address the limitations of the preliminary socio-technical congruence model forms the basis for the empirical investigations described in this dissertation.

Chapter 3

A Model of Socio-technical Congruence

In order to study socio-technical coordination within a global software team, a technique is needed to conceptualize and measure both technical relationships and social relationships. Socio-technical congruence allows a researcher to represent coordination within a software-development organization in a quantitative manner. It reflects the earlier sensibility of Conway's Law, which is the observation that a software's architecture reflects the structure of the organization that builds it [56]. By mapping the relationship of a person to a technical artifact, such as source code, and the dependencies between these artifacts, one can form a network of relationships that identify the coordination between individuals in order to measure coordination and achieve good performance.

> **Objective 1: To develop a socio-technical congruence model that explains what conceptualizations belong in the model, what elements belong in the model, and how to calculate alignment.**

To achieve this objective, I investigated related work and identified aspects in coordination and socio-technical congruence to develop this model.

3.1 The Purpose of a Socio-technical Congruence Model

The purpose of a model of socio-technical congruence is to enable a researcher to study socio-technical coordination by identifying the relationships between people, as well as the alignment between their social behaviour and their technical work.

This model expands on the socio-technical congruence concept introduced by Cataldo [44]. Cataldo in his dissertation concentrated on examining dependencies and work assignments needed to calculate socio-technical congruence rather than describing conceptualizations and guidelines for applying the technique. Sarma et

al. [187] came closest to describing a process for socio-technical congruence by highlighting conceptual situations that congruence can be applied to and by illustrating ideas for how to implement changes that achieve congruence.

Because of the diverse number of studies and conceptualizations within socio-technical congruence studies, a model provides a consistent perspective of what is included in a socio-technical congruence study, as well as common language. It also assists researchers when designing studies and interpreting results.

3.2 Related Work in Socio-technical Congruence

Current research suggests that attaining a high level of socio-technical congruence is beneficial to an organization. Evidence shows that higher congruence leads to faster completion of modification requests [50]. The presence of gaps increases the number of code changes [84], and a lack of coordination connections across system and organizational boundaries have a negative effect on performance [200].

Much of the work thus far on socio-technical congruence is somewhat fragmented; the existing socio-technical congruence workshops have featured papers that range from analysis of architectures and organizations to social structures based on social network analysis. In this dissertation I am primarily concerned with conceptualizations of technical relationships and social relationships at the task level such that a measurement of congruence can be calculated.

Cataldo et al. [50] evaluated congruence in the context of software development using different representations of social relationships, including geographical proximity, IRC communication, and issue-tracking comments; these factors correlate with the resolution time of modification requests. He conceptualized a technical relationship as someone who contributed to a file that was a part of a modification request. In another investigation, Cataldo et al. [48] examined different types of technical dependencies by using differing ways to measure architectural dependencies. The authors found that the congruence values computed using a logical "files changed together" dependency [98] were more reliable than syntactic call graph dependencies [78]. In the automotive domain, Gokpinar et al. [101] applied a congruence technique and discovered that a higher coordination deficit led to a larger number of filed incident reports, implying reduced quality.

Cataldo et al. [49] also investigated software failures in relationship to different kinds of work dependencies, discovering that an increase in the number of logical dependencies, as well as workflow dependencies that represent the transfer of a task from one developer to another, were likely to lead to modifications required as a result of a failure. However, the technical relationships, conceptualized as "the most central developer in a coordination requirements developer-developer matrix that worked on the file i" did not have as strong an effect as other types of dependencies on quality. Intuitively, the coordination requirement represents the developer working on a file who was connected to the largest number of other developers given a definition of "file

i and file j were modified together". While the study did not investigate alignment, it revealed insight into the usefulness of technical dependency conceptualizations.

Socio-technical gaps are an issue not only because they lower the congruence and thus lower productivity [50], but because they are especially problematic in the context of distributed development [84]. Thus, researchers have proposed remedial actions when socio-technical congruence gaps are discovered [211]. Examples of actions include closing a gap by augmenting coordination and eliminating the gap by refactoring software.

Socio-technical congruence has been explored outside of the software development field, particularly in engineering and management disciplines [110, 200, 101, 199]. Sosa [200, 199] described a technique to compute socio-technical congruence and classified gaps into categories such as "potentially unattended technical interactions". Gokpinar [101], independently of this dissertation's research, developed a weighted socio-technical congruence technique that was applied to the automotive industry. The socio-technical congruence concept has been used in the diagnosis of coordination issues [16] as well as in expertise recommenders [157].

Some proponents states that not only does alignment happen as a consequence of product architecture (as in Conway's Law), but that it is in fact a desired property, and propose methods to attain alignment [16, 187, 211]. In general, the rationale is that coordination can be expensive, especially over distance, and thus coordination should be performed in a way that minimizes unnecessary coordination while still ensuring that there is enough to complete tasks [114]. A description of ongoing challenges in socio-technical congruence are listed by Sarma et al. [187]. In their position paper, congruence is described as a state, as dynamic, and as multi-dimensional. Examples of data to investigate and project performance are proposed, along with a research agenda to understanding congruence, which describes the need to build theories, relate congruence to organizations, understand paradigms of congruence, understand effects, and to design metrics and visualizations of socio-technical congruence.

3.3 Limitations of Socio-technical Congruence

There are a number of limitations with the current conceptualizations of socio-technical congruence based on an investigation of related work. These two limitations, which I propose solutions for in the model, are the *binary nature of gaps* and the *difficulty of comparing socio-technical congruence studies*.

A gap in socio-technical congruence is represented as a binary relationship. The socio-technical congruence measure identifies when people's technical relationships are satisfied by their social relationships. There may be multiple technical relationships that exist between two people in a project, but the current socio-technical congruence measurement considers a single social relationship as being sufficient to

cover every technical relationship. Another problem is the inability to identify "partial" gaps, where some of the technical relationships are satisfied, but others are not.

Previous research has shown at least one case in which a project succeeded despite the presence of gaps [151]. Ehrlich et al. [84] suggest that having an individual act as an intermediary between two groups, known as a *broker*, may be able to mitigate the effect of gaps. Communication structures may compensate for a lack of congruence [118]. Literature in areas such as management science supports the idea that gaps are not a liability, but simply a reality [78, 120].

As developers have multiple dependencies [76, 159], one would presume that those pairs with more technical relationships would need to coordinate more. Current socio-technical congruence provides no way to identify which pairs have high technical relationships. Researchers have suggested that both social and technical dependencies should be weighted [212, 139], but do not evaluate their proposed changes.

Socio-technical congruence studies are difficult to compare. Another current challenge in evaluating the effects of socio-technical congruence is that the large number of elements and entities make the comparison of studies difficult. This is not an inherent limitation of socio-technical congruence, per say, but is an issue when trying to understand the overall effects of the socio-technical congruence.

Socio-technical congruence has been used to study the effect on response variables such as "defect resolution time" [50, 48], conceptualized as the amount of time that a defect in an issue-tracking system is open; "number of code changes" [84]; and number of warranty incidents [101]. The issue with defining performance is that it is context-specific [153], and consequently the appropriateness of whatever metric is used depends highly on the context of the case to be studied.

This issue of comparability is further compounded by the fact that many of the conceptualizations within studies use different entities for analysis. Some conceptualizations rely heavily on extracted data from repositories (ex: [50, 48]) whereas other studies of congruence have relied on questionnaire data [84, 152]. While there is no "right" set of conceptualizations to use, clearly showing the representations used assists researchers wanting to learn from these empirical studies of socio-technical coordination.

Using identical response variables with identical entities is neither realistic nor beneficial to research. What is needed instead is a way to quickly identify what the appropriate elements taken from each empirical study are such that a reader can quickly form a mental model of the variables that are represented. A socio-technical congruence model can help identify the common traits between different studies and provides a way to guide the selection of conceptualizations. The model can highlight strengths and weaknesses of the choice of conceptualizations. In general, the model provides a common language that can be used when examining studies that use socio-technical congruence.

3.4 A Socio-technical Congruence Model

The main idea around socio-technical congruence is that it involves technical relationships and social relationships and that the technical and social relationships align with one-another. If there are not enough social relationships to match the technical relationships, then congruence is less than 1. In addition to identifying the effect of socio-technical congruence on a software-engineering outcome, analyzing these networks and identifying gaps permits exploration into potential problem areas of the network.

Socio-technical congruence uses a number of social networks. A social actor is usually a representation of a person, though it may also represent a group. More information about social networks is described in Section 2.3.4. For simplicity, I consider relationships between people only, except when explicitly stating otherwise.

A socio-technical congruence model at least requires two different actors: one type of actor that represents a person or an organization for the social side of the organization, and one type of actor that represents the technical side of the organization. I call the actor that represents the person or organization a **social actor** and I call the actor that represents the technical side of the organization a **technical entity**.

Thus, socio-technical congruence can be viewed as the overlap between a bipartite social network that containing social actors and technical entities, and another social network that contains social actors only.

3.4.1 Model Elements

In addition to the primary entities in the social networks, the socio-technical congruence model consists of three elements illustrated in Figure 3.1: *conceptualizations* of the relationships and entities in the model (Section 3.5), *techniques* for calculating the alignment and the relationships in the model (Section 3.6), and guidelines on how use the model to represent *exceptional situations* (Section 3.7). For simplicity, I call each of these "conceptualizations", "techniques", and "situation modelling".

The interaction between these parts in the socio-technical congruence model appear in Figure 3.2. The model is characterized by the conceptualization of social relationships between people and the conceptualization of technical relationships between people. These elements are connected to each other, and are required in order to calculate alignment, using one of the alignment techniques. These in combination make up socio-technical congruence, which represents social relationships and technical relationships, as well as their alignment.

The alignment value can be used as an input variable to study a statistical relationship to a response variable, or it can be used to improve understanding of situations within the software organization. A response variable is a variable that is of interest in a software-engineering domain—for example, "mean time to failure", "defect resolution time", "probability of build success", and so on.

Technical relationships and social relationships are interrelated, and thus it is

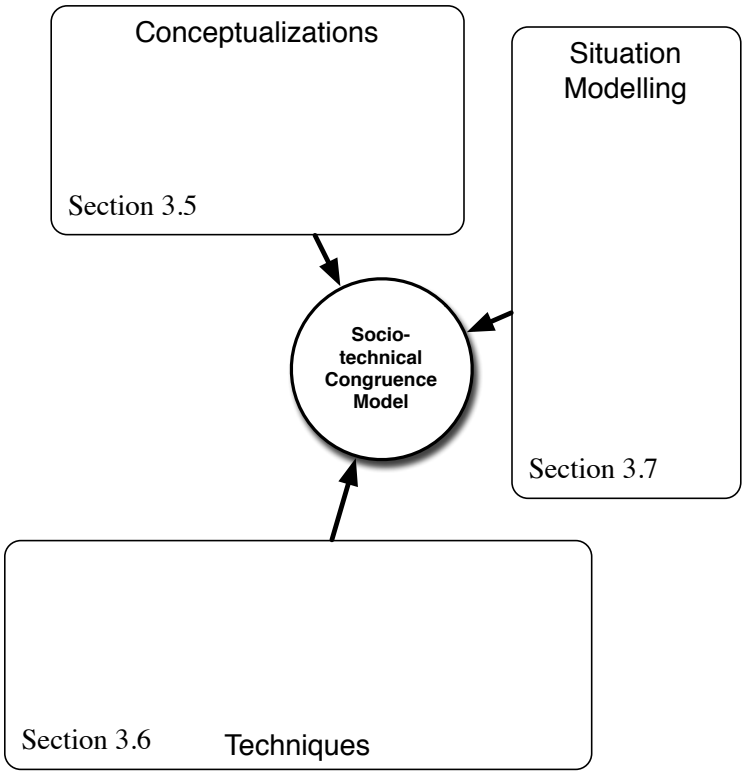


Figure 3.1: Elements of the Socio-technical Congruence Model

beneficial to consider both in parallel. For the purposes of presentation, I describe the conceptualization of technical relationships first, and social relationships second, though the conceptualization should occur interchangeably to achieve alignment.

Figure 3.1 forms a foundation for the areas of investigation. Each empirical investigation in this dissertation addresses different parts within each element of the model.

3.5 Conceptualizations of Relationships

Conceptualization refers to the representation of the people and artifacts from a software development team in a socio-technical congruence model. Within the conceptualization part, one must choose how to represent technical relationships and social relationships. These relationships should be chosen such that the social relationships are needed or desired in order to execute work identified by technical relationships.

Cataldo et al. [48] calls the relationships in the technical network "coordination requirements" and the relationships in the coordination network "actual coordination". However, I felt that these labels were potentially confusing[1]. To make the

[1]In particular, "actual coordination" cannot be distinguished between a singular and a plural form, which can cause ambiguities. See Berry and Kamsties [25].

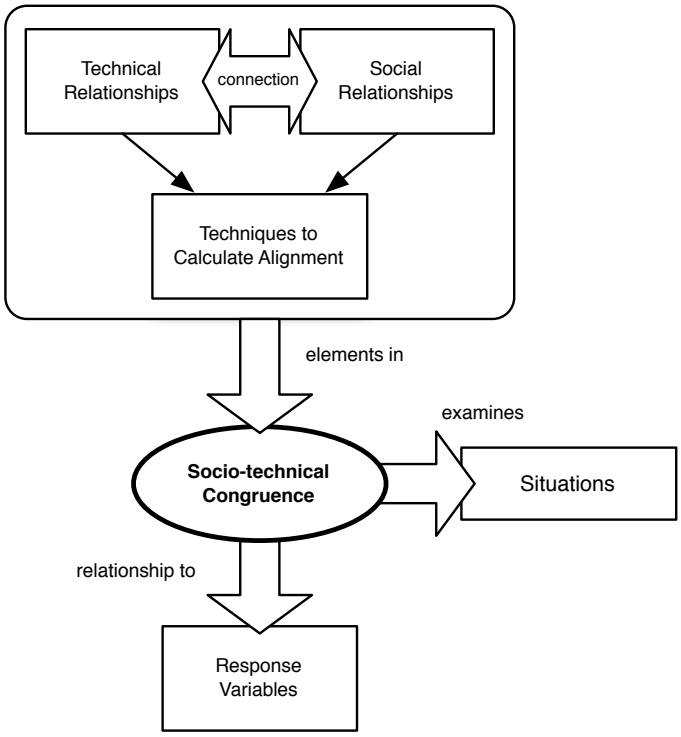


Figure 3.2: Overview of the Socio-technical Congruence Model

concepts in socio-technical congruence more intuitive, and to emphasize the contrast between the coordination requirements derived from technical dependencies, and the actual coordination recorded from the behaviour of team members, I rename "coordination requirement" to **technical relationship** and "actual coordination" to **social relationship**.

Because socio-technical congruence involves the alignment between technical relationships and social relationships, there are at least two conceptualizations that a researcher must apply: the technical relationships, and the social relationships. Examples of some commonly-applied relationships appear in Figure 3.3. I also illustrate how these examples are a close-up view of the conceptualizations box in the model overview (Figure 3.4). I also propose a third entity called a **connecting entity** that connects the social and technical aspects to ensure that the social relationships satisfy the technical relationships.

3.5.1 Technical Relationships

A **technical relationship** is a relationship that indicates that one person is dependent on another person. Before a technical relationship can be derived, both a technical entity and a technical dependency must be conceptualized. Both Cataldo [44] as well as Valetto et al. [212] describe the technique of forming technical relationships, with Cataldo using matrix algebra and Valetto et al. using a graph-based algorithm.

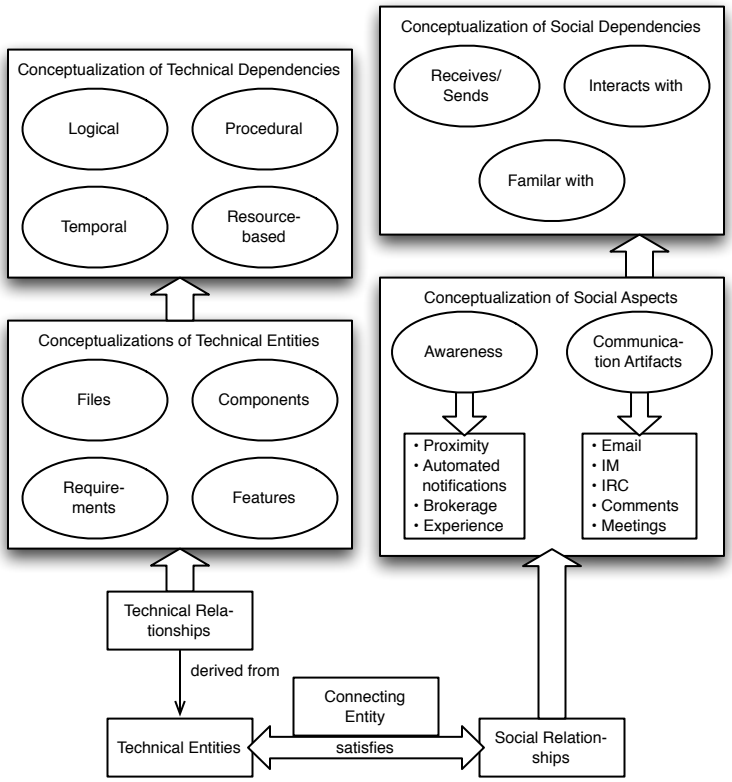


Figure 3.3: Examples of Conceptualizations for Technical Relationships and Social Relationships

A **technical entity** is an entity in a project that can be worked on by a person or a group. Examples of a technical entity include a source code file, a compiled binary, a requirement, a task, or a bug. Socio-technical congruence has focused on the source code file as the technical entity [50, 84], although work has also examined socio-technical congruence using a requirement [70, 152] or a task [224] as the technical entity.

A **technical dependency** is a type of dependency between two technical entities. These relationships are the dependencies of socio-technical congruence and are discussed at length by Cataldo [44] in his dissertation. Additional mentions of dependencies include requirements dependencies by Dahlstedt and Persson [68], logical file dependencies by Gall et al. [98], API calls [201], and topic modelling [171, 117]. Examples of technical dependencies are in Table 3.1.

A **technical assignment** is a relationship between a technical entity and a person. In most situations this is simply an assignment, perhaps from a project plan or a bug-tracking database, though it may also be conceptualized as "last modified" or "worked on most frequently". This relationship is called a "task assignment" by Cataldo [44].

Technical relationships can be viewed as a bipartite network made up of two types of actors: the people in the organization, and the technical entities. Connections

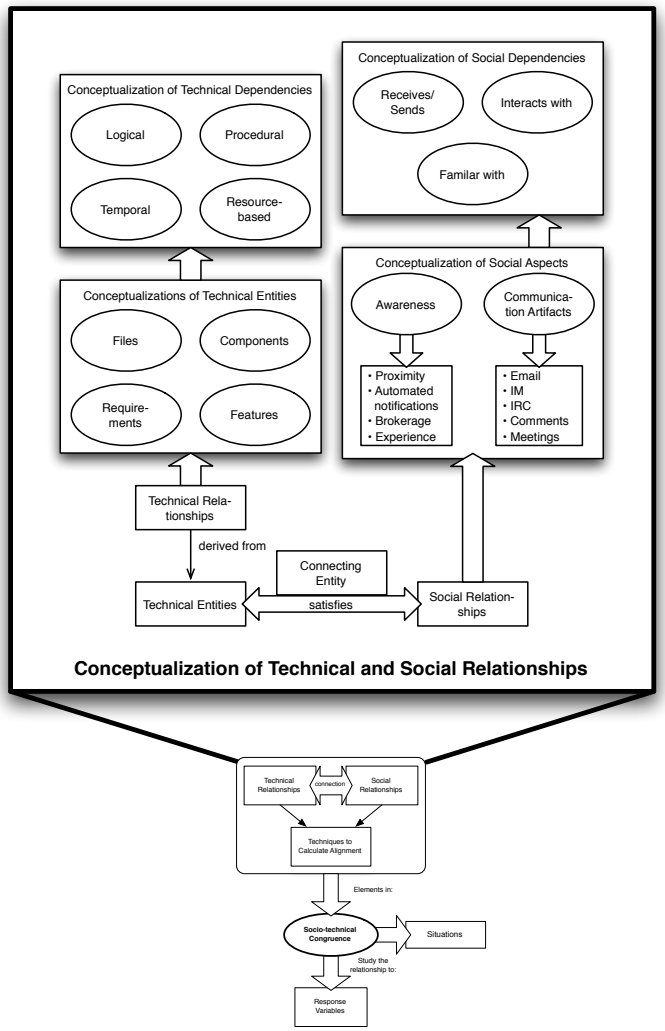


Figure 3.4: Zooming into the Conceptualizations Block in the Socio-technical Congruence Model

between an entity and another entity are technical dependencies, connections between an entity and a person is a technical assignment, and connections between a person and another person are technical relationships.

Calculating Technical Relationships Socio-technical congruence uses a mathematical representation called an adjacency matrix to indicate relationships between people. An adjacency matrix A is of size $m \times m$, where m is the number of people in the network. The adjacency matrix has elements

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge between persons } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases}$$

The adjacency matrix use the technical relationships conceptualization from Sec-

Entity	Dependency
Requirements	Interdependency [151, 152]
Source code modules	Changed together in a change set [50, 48]
Source code file	Call graph dependency [78]

Table 3.1: Examples of Technical Dependencies

Relationship	Description
Communication	A communicates with B [50, 84, 48, 72]
Location	A is in the same location as B [50, 84]
Team structure	A is in the same team as B [50]

Table 3.2: Examples of Social Relationships

tion 3.5.1 and and the social relationship from Section 3.5.2 as part of the calculations. While I use matrices in this thesis to represent calculations, an alternative graph-based algorithmic method has also been proposed [212]. One can also use edge list representations as well, where an edge is represented with a $source - - - target$ representation.

Given m people, and n technical entities to examine, the technical relationships are calculated as

$$TR = TA \times TD \times (TA)^t \tag{3.1}$$

where TA is an $m \times n$ technical assignment matrix that indicates that a person is assigned to a technical entity of the project, and TD is a $n \times n$ technical dependency matrix that indicates that a technical entity is dependent on another technical entity [50]. The result is TR, an $m \times m$ technical relationship matrix that indicates who in the project should be coordinating with whom.

3.5.2 Social Relationships

The second kind of relationship in socio-technical congruence is the social relationship. A social relationship can be either calculated based on artifacts, or identified directly, perhaps through observational techniques. If artifacts are used for communication, the relationship between artifacts and people must be defined before a social relationship between two people is established, similarly to how technical assignments are used to calculate technical relationships. Some examples of social relationships appear in Table 3.2.

Direct Social Relationships

Many social relationships can be modelled directly. For example, if one were to represent "familiarity" between two people, no artifacts are involved—the people are simply familiar with each other directly and therefore are connected. This data may be acquired using a questionnaire or an interview.

Other examples of a direct relationship would be an organizational relationship that is extracted from an organizational chart, or whether two people work in the same office building.

Social Relationships via Artifacts

Social relationships can be extracted by using artifacts. These artifacts are most often communication artifacts, such as email, instant messenger, or IRC. Unlike a direct relationship, some intermediate relationships need to be conceptualized in order to apply these artifacts to socio-technical congruence. Most often, communication artifacts come from repository mining.

The artifact relationship requires two endpoints: a **source**, and a **target**. This is usually resolved by a "sender–receiver" relationship, or a "creator–receiver" relationship. The source and target tend to be relatively intuitive, but there are exceptions, including non-directed communication like forums and mailing lists where the target relationship is difficult to identify.

In a shared workspace, an "accessed" field that indicates who has accessed the item is probably the best way to know that someone has accessed the artifact, but most workspaces do not track this. Thus, another technique must be used to estimate someone's reception of a message.

Thus, if there are social artifacts being used, a matrix SR is calculated using Equation 3.2, where SA is a matrix indicating a source i is linked to artifact j matrix, and AT is an artifact j linked to target k matrix.

$$SR = SA \times AT \tag{3.2}$$

In a shared workspace that involves synchronous communication, such as IRC, one way to model communication artifact targets is to simply indicate that everyone present in the system receives the message. Thus, if five people are in an IRC chat room and one person sends a message, it is assumed that every person currently in the chat reads the message. This of course relies on the assumption that the participants are actually monitoring the chat and are not away from their keyboards. To get around this, a common technique is to model the target as a "replier" to the first message. This usually occurs as follows. When the source sends a message to a workspace, and someone creates a new message and posts it in reply to the first message, the initial message creator is the source, and the new message creator is the target. It is assumed that the new message creator read the first message and then was so invested in the message that he crafts a new message and replies—such an assumption has been made by Bird et al. [27] and Nguyen et al. [166] in the past.

This technique tends to model participation well but also underestimates the number of people who read the first message.

The choice of the social relationships also needs to reflect the technical relationships. As the purpose of socio-technical congruence is to identify how social relationships align with technical relationships, there should be a way that the communication artifacts align with the technical relationships. Most existing socio-technical congruence studies use artifacts that are logically-related to the technical entities through explicit connections, though some have used manual verification to check that communication topics align with the technical artifact topics.

3.5.3 Connecting Entity

A connecting entity is an entity that identifies the link between the technical relationships and the social relationships. This entity is necessary to consider because many software organizations do not record the relationship of social relationships and technical relationships directly. For example, the mapping between social relationships, such as an email message, and its relevance to the work that two people have together is not necessarily explicit.

Most existing socio-technical congruence studies do not explicitly define this entity as something that requires conceptualization. However, because of the diversity of data collection techniques it is important to explicitly express the traceable link between the social relationship and the technical relationship that the social relationship is supposed to satisfy.

An example of a connecting entity is a logical grouping of files, such as a modification request, on which developers can leave comments. Another connecting entity could be a questionnaire that asks if a team member works with another on a specific technical entity.

3.5.4 The Scope of a Socio-technical Network

A network has a limited number of entities. There are a number of ways to identify which elements belong in which networks. The most common classifications are temporal networks, in which an actor or entity is included in the network if either the entity or the actor has some action executed on it during an arbitrary time period. An example of this appears in [121]. The other classification is a logical grouping of entities. This occurs when the technical entities are included within a larger entity. Most commonly, this larger entity is the "project release", and if a file is a part of the project, then it is included in the network. An example of this situation appears in Cataldo et al. [50].

As mentioned by Sarma et al. [187], congruence is a *snapshot* at one point in time. It reflects the work done during that period. Because congruence is snapshot, the expected situation is that there is no congruence at the start of a cycle, and as the cycle advances, congruence generally increases. To counter this, Wagstrom et

al. [216] proposed a way to compute an inherent delay in socio-technical congruence.

3.6 Alignment Techniques

Alignment techniques refers to calculations that compute the "fit" between the social relationships and the technical relationships. Techniques include the standard congruence calculation using a ratio, but also include more elaborate calculations like decay factors in socio-technical congruence [216], using weighted relationships [48, 139, 101], and variable gap sizes [139].

In this section, I devote attention to the existing calculations of alignment. I also propose my weighted congruence measure [139] that addresses some of the limitations of the original alignment technique.

3.6.1 Calculating Socio-technical Congruence

Given a technical relationship matrix and a social relationship matrix, congruence is calculated by comparing this matrix to a social relationship matrix. If an edge exists in the social relationship matrix, and the same edge exists in the technical relationship matrix, then there is congruence on this edge. Given definitions

$$\text{Diff}(TR, SR) = card\{\text{diff}_{ij} | TR_{ij} > 0 \ \& \ SR_{ij} > 0\}$$
$$|SR| = card\{SR_{ij} > 0\}$$

the socio-technical congruence index is calculated as [48]

$$\text{congruence} = \frac{\text{Diff}(TR, SR)}{|TR|} \tag{3.3}$$

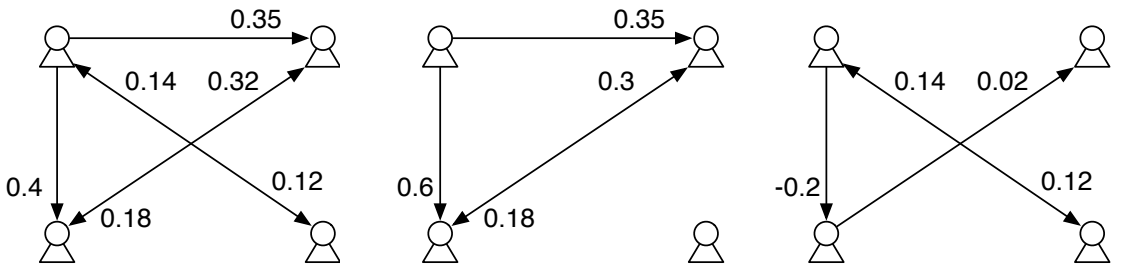
where SR is an $m \times m$ matrix representing social relationships. The resulting congruence value is a number between 0 and 1, where 0 indicates no congruence, and 1 indicates full congruence.

3.6.2 A Weighted Congruence Measurement

In this section, I present an extension to congruence based on prior work [139] that addresses the gap size limitation described in Section 3.3. Weighted congruence addresses the problem by allowing multiple technical relationships and multiple social relationships. The goal of this improvement is to allow a researcher to specify additional information about coordination behaviour and to assign priority to gaps.

Weighted Technical Relationships

This measurement assigns a number between 0 and 1 to each entry in the technical assignment, technical dependency, and social relationship matrices. The weighted



(a) The Technical Relationships (b) The Social Relationships (c) The coordination that is Lacking

Figure 3.5: Comparing Weighted Technical Relationships with Weighted Social Relationships to Find Lacking Coordination

technical assignment matrix is a $m \times n$ matrix where m is the number of people and n denotes the number of entities. Each entry in the matrix illustrates the strength of the connection between a person i and an entity j. Such a connection may be the expertise person i has with a task j, or the priority that person i places on task j.

The weighted technical dependency matrix is an $n \times n$ matrix where n denotes the number of entities. Each entry in the matrix describes the strength of the relationship between two entities. The weighted technical dependency matrix may show the ratio of dependencies an entity has with another over the number of total dependencies that entity has on others. For example, an entity strongly coupled with another has a high entity dependency.

To calculate the technical relationships matrix, I use Formula 3.1, using the weighted values in the technical assignment TA' and technical dependency TD' matrices.

$$TR' = TA' \times TD' \times (TA')^t \tag{3.4}$$

This calculation will yield the weighted technical relationship matrix TR', an $m \times m$ people by people matrix (Figure 3.5(a)). The technical relationship matrix shows how strong the relations between people are supposed to be based on their technical work.

Using weighted congruence allows the opportunity to represent technical relationships in a manner that is more precise than previous relationships. The primary change is that a connection between the same two people, but through different artifacts, can be represented as a different technical relationship. In a software development team that engages in relatively fine-grained communication, each of these technical relationships may need to be addressed by individual coordination instances.

Weighted Social Relationships

A weighted social relationship is a relative value that indicates the strength of a relationship between two individuals (Figure 3.5(b)). A social network such as a commu-

Variable	Description
Syntactic dependencies	Number of dependencies
Meetings	Number of meetings in a time period with others
Distance	Physical distance (in metres)
Email messages	Number of email messages
Discussion threads	Number of threads related to the technical topic that a person participated in

Table 3.3: Examples of Relationships Using Weights

nication network can be weighed according to the amount of ongoing communication. For example, one such weighing scheme may work as follows:

1. For a communication network, identify the largest value of communication between two individuals.
2. Divide all other values by the largest, thus normalizing the communication.
3. The result is a number between 0 and 1, where 1 indicates the largest value of communication in the network, and 0 is no communication.

The weighted calculations are applied similar to Equation 3.4 if one is using social artifacts to represent social relationships.

Some examples of weighing relationships appear in Table 3.3.

3.6.3 Identifying Gap Size

I complement my weighted congruence measure with a technique to generate a *lack-of-coordination matrix*. This matrix not only identifies the gaps between a technical relationship matrix and a social relationship matrix, but can also identify the size of these gaps (Figure 3.5(c)). This lack-of-coordination matrix can be used with both unweighted congruence and weighted congruence.

$$g_{ij}(TR'_{ij}, SR'_{ij}) = TR'_{ij} - SR'_{ij}$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, m$. The value of g_{ij} is the **gap size** between the pair ij. The lack-of-coordination matrix illustrates situations where the proportion of communication exceeds the expected proportion of technical relationships. If the value is positive, then there is not enough coordination to satisfy the information needs requested by the TR' matrix, resulting in a **gap**. If a value is negative, then there was more coordination than requested through the TR' matrix, resulting in **overload**. Neither situation necessarily results in a problem, but I believe that investigating overload and lack of coordination has implications on software engineering coordination.

3.6.4 Weighted Congruence Index

To convert the lack-of-coordination matrix to a single socio-technical congruence measurement, I apply

$$\text{congruence} = 1 - \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} g_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{m} TR'_{ij}}$$

for each i and each j, unless $i = j$. This value, which is usually between 0 and 1, is an overall level of the congruence in the current network. In theory, the formula allows the index to fall outside of this range for some edge cases, but I did not observe any such cases in practice.

3.6.5 Benefits of Weighted Congruence

Using a weighted congruence model allows us to deal with situations that are not handled with unweighted congruence. Using the weighted congruence approach allows a person diagnosing the organization to benefit from *locality* and *relationship strength*.

Locality allows us to identify which area of a network has a gap. The lack-of-coordination matrix identifies which pairs of individuals do not fulfill technical relationships. Although the original model does give us some limited locality in the sense of identifying congruence gaps, it is far more coarse grained.

Relationship strength indicates how strong the relationship between two developers is. Relationship strength allows us to identify pairs of people who have multiple technical relationships, and therefore who need to coordinate more often with each other. Using relationship strength allows us to investigate coordination in more detail because I can investigate pairs with strong dependencies—these people also tend to be key individuals in a team.

By allowing better locality and relationship strength, I believe that this weighted model provides an in-depth technique with which to study congruence.

There are a number of extant techniques for modelling multiple relationships in literature, which I present in the following section.

3.6.6 Comparison with Other Weighted Measures

There are a number of other methods for weighing individual socio-technical relationships in literature. Cataldo proposed a system using weights with integers [48]. Ehrlich et al. [84] used a method to rank communication into three levels of importance. Gokpinar [101] independently proposed a weighted congruence measure. I briefly contrast the different types of weighted congruence below.

Integer Matrices

The method of Cataldo et al. [48] assigns integer weights as values in the task assignment and task dependency matrices, resulting in a weighted technical relationships matrix. However, his formula in Section 3.6.1 considers that a single communication instance is sufficient for satisfying any amount of coordination. Our method provides information about overload and lack of coordination, and provides a more fine-grained representation through the lack-of-coordination matrix.

Levels of Communication

The method of Ehrlich et al. [84] aggregates multiple relationships between two individuals to determine a "level of communication". The relationships are: when two individuals work together on a project; when two individuals communicate several times a month for any reason; and when two individuals work together on shared files. If one or more of these relationships are true, they are added to provide a number representing the level.

The level of communication does not provide a ranking of gap size because they presume that a gap either exists, or that it is covered with one of the three levels. This paper mentions ranking gaps by size, but the gap rank is the total number of gaps that exist between two developers over the entire project. Thus, this measurement does not actually weigh the conceptual lack-of-communication distance between two individuals for each gap.

Coordination Deficit

Gopkinar et al. [101] independently proposed a weighted congruence called "coordination deficit", which is equivalent to my gap size measure. It weighs the number of dependencies between components, and the number of communications between individuals. Like our method presented in prior work [139], they calculated the coordination deficit by normalizing the edges and subtracting the values between the equivalent of a social relationships network and a technical relationships network. The fact that the authors used normalization and algebraic subtraction suggests that they are reasonable approaches to calculating gap size.

3.7 Situation Modelling

Situation modelling refers to identifying exceptional situations that may appear in socio-technical congruence settings. The exceptional situations, identified from related work, are important communicators in the network, and emergent people. These situations do not fit into the definition of "conceptualization" outlined above, and instead are a consequence of the analysis rather than as a consequence of conceptualization.

One possible solution for these situations is simply to migrate these emergent people toward the next "snapshot" of socio-technical congruence according to Sarma et al's definition of snapshot [187]. If they are communicating about technical topics then it is possible that these emergent people are technically dependent on these topics.

3.8 Using the Model: A Socio-technical Congruence Study Template

In order to guide the construction of a socio-technical congruence study, I invented a template within which a study's conceptualizations can be placed in Figure 3.6. A tabular version of this template appears in Table 3.4.

The top of the template is descriptive and requests a project name as well as a summary of the data available. The centre of the template indicates the alignment between the technical relationships as well as the social relationships. The technical relationships half requests information about the entity, the entity's connection to people, and the technical dependency. The social relationship half requests information either about the social artifact, or a description of the social relationship. In the social artifact field, there is space to describe the source and the target (if needed). The social relationship has a field that describes how each satisfies the technical relationship. If necessary, either technical relationships or social relationships may be duplicated as needed to represent the situation. The connecting entity should also be filled out to indicate the map between the social and technical relationships.

The alignment techniques used are described in a field. Usually, this is the basic "alignment" calculation (Equation 3.3) but can include weights. The bottom of the template describes the output of applying socio-technical congruence. There is one box for describing the analysis method (such as statistical analysis, network analysis, etc.) and the response variable of the study.

By placing the techniques, conceptualizations, and the response variable in the same template, one can quickly inspect the conceptualizations and can identify the strengths and weaknesses of the conceptualizations used, especially with respect to the alignment between the social and technical relationships. This can be used to compare different socio-technical congruence studies, or to ensure that the data exists for collecting each piece of data.

3.8.1 Application of the Template

I tested the socio-technical congruence template by applying it to previous socio-technical congruence studies in literature. Below, I show the template for two of the congruence empirical investigations.

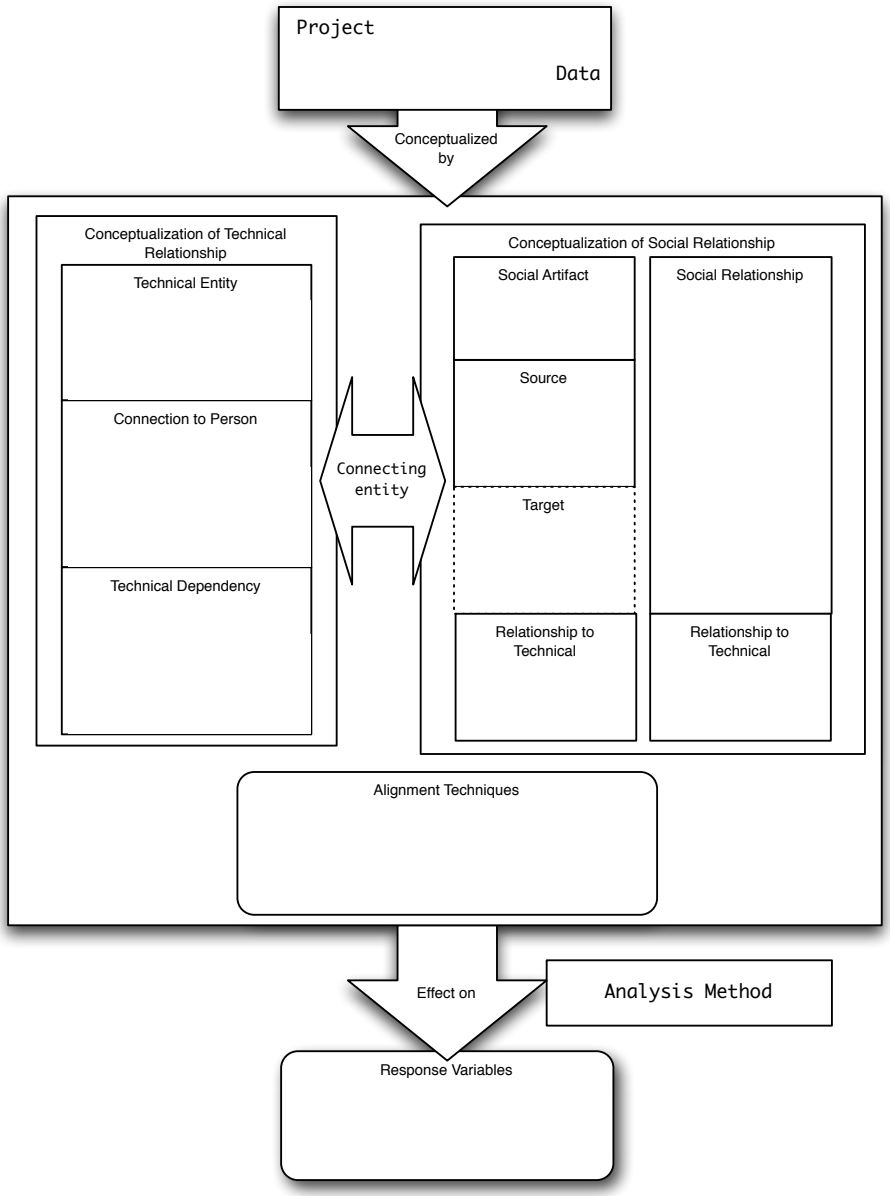


Figure 3.6: The Socio-technical Congruence Study Template

Category	Description
Study Data	
Technical entity Technical dependency Technical assignment	
Connecting entity	
Social artifact entity Social artifact source Social artifact target Social artifact relationship to technical	
Social relationship Social relationship to technical	
Alignment techniques	
Analysis method Response variable	

Table 3.4: Tabular Version of Socio-technical Congruence Study Template

Data Storage Case Study

Cataldo et al. [50] described a setting of a distributed company within the data storage industry. The completed template appears in Figure 3.7.

Key areas of note in this template is the use of a modification request (MR) as the connecting bridge between the technical relationships and the social relationships. A modification request contains a number of modified files (indicated in the technical relationship section) as well as number of comments (indicated in the social artifact section). The authors also explored alignment between the social aspects using formal team structure, and geographical location. They also examined IRC logs and identified when a developer mentions the MR ID or a similar task/problem in chat. Because this study utilized multiple social relationships, the social relationship "box" was expanded to fill in the extra details.

At a glance, this template provides a view of the variables for study, as well as the relationships between these different entities.

Learning Management System Case Study

The learning management empirical investigation described by Ehrlich et al. [84] is a distributed project that builds a web-based system for learning management. The goal of this study was to examine gaps. The completed template for this study appears in Figure 3.8.

The conceptualizations in this study are not as closely aligned as in the Data Storage empirical investigation. There is a lack of a central connecting entity—rather, the authors rely on different sets of social relationships. They do not directly align

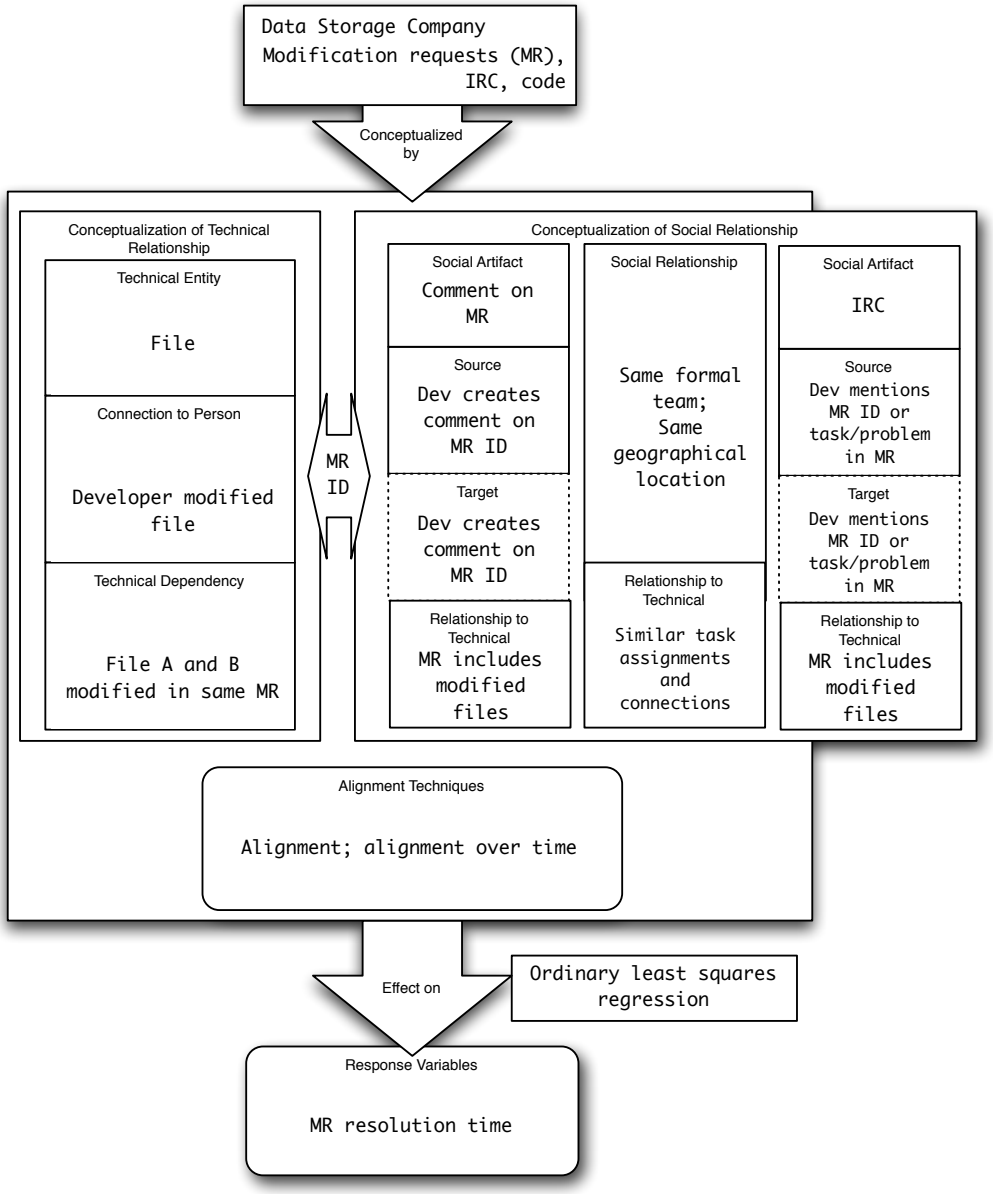


Figure 3.7: Socio-technical Template Applied to Data Storage Company Described by Cataldo et al. [50]

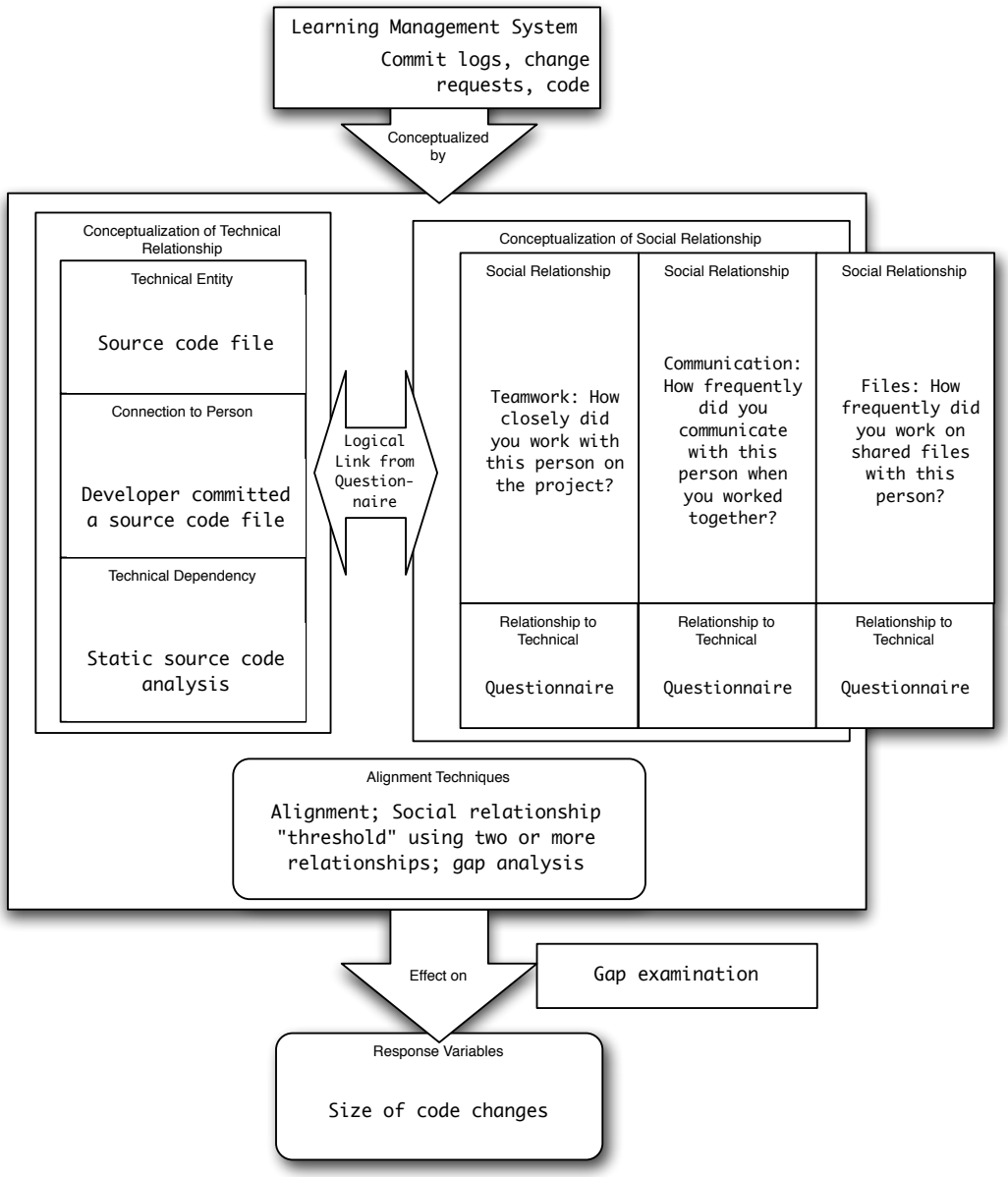


Figure 3.8: Socio-technical Template Applied to Learning Management Empirical Investigation Described by Ehrlich et al. [84]

these relationships with the technical entities. However, to avoid underestimating the amount of communication, they do not indicate a relationship unless two or more of the social relationships exist.

One detail that was not made explicit in this empirical investigation is the link between the social and technical relationships. The study state that a questionnaire was sent to team members, but no details about whether the questions are specific to particular software subsystems are provided.

Thus, the template again allows an at-a-glance view between the entities that are studied using STC, and can quickly highlight the benefits and drawbacks of various choices of entities and relationships.

3.9 Missing Elements from the Socio-technical Congruence Model

Though I have introduced some solutions to conceptualizing and representing relationships, as well as a technique to address gaps, there are still some outstanding issues. Early investigations into socio-technical coordination examined awareness and communication "in the wild" within an industrial software team [69]; both awareness and emergent people came up in this study as concepts that were not well-studied in socio-technical coordination. While awareness comes up primarily in the computer-supported collaborative work domain [105, 164, 205], it is not as well described within a socio-technical environment featuring a design team, such as a software-engineering team [39].

An enhanced understanding of the social interactions with a global software team would benefit not only team performance, but also the conceptualizations of these interactions within socio-technical congruence. By enhancing the applicability of socio-technical congruence to multiple scenarios and empirical investigations in global software development, we are providing a flexible tool for the study of socio-technical coordination.

There are three specific limitations with existing socio-technical congruence conceptualizations that can be investigated.

The influence of awareness on a team's technical work, in the context of socio-technical congruence, is not well-understood. There is evidence that common representations of social relationships, such as communication, are not required to satisfy technical relationships. Whereas Conway's Law [56] states that the software architecture will reflect the organization that built it, socio-technical congruence takes this assertion and assumes that it is a recommended best practice. There are some issues with this assumption. The most prominent one is the fact that it is possible for people to work on software artifacts without explicit coordination [201]. Bolici et al. [34] observe two open-source projects in which people do not communicate directly. Software is designed to be extremely modular, with components and

APIs [78]. With good APIs, the need for direct communication is eliminated. Documentation may also help developers coordinate. Though documents are a form of communication, it is difficult to record the usage of documentation as communication from one person to another. Coordination does not necessarily require the involvement of the specific individuals who are working on the task. If someone else in the organization is familiar with the task, then that person may be able to provide expertise. A study of brokerage in software engineering [151] identified individuals who communicate on behalf of others.

There are multiple methods to coordinate without communication. Awareness variables in literature have been used to conceptualize social relationships as well. Organizational structure [50], geographical distance [50, 72], and the person's role [152] have each been conceptualized as affecting social relationships conceptualizations. However, no single technique attempts to identify the effect of each of these interactions on software development. Exploration into how team members become aware of each others' tasks, and how this awareness affects their work is warranted. Thus, this leads us to Objective 2 of the thesis.

> **Objective 2: To examine awareness and to identify how awareness within a global software-development team can be conceptualized within a socio-technical congruence model.**

Current conceptualizations of socio-technical congruence do not always involve every relevant team member. Important communicators and people who are involved in a conversation after the conversation starts are not included in most socio-technical congruence conceptualizations. Because congruence uses the technical relationships as a base, it means that people who are involved in a project primarily through social relationships are not considered as "part of the task". Some roles in socio-technical congruence are not well-represented. For instance, Marczak et al. [152] could not represent the project manager because he had no explicit dependencies to requirements. Another communication pattern that has been identified is the emergent people pattern [69, 72]. The emergent people pattern is a situation that describes when more people are involved in communication than initially expected. Socio-technical congruence does not provide any options for handling emergent people and currently ignores them. Some more knowledge about communication, and how it reflects technical relationships, is needed, especially with respect to identifying who the prominent communicators are in a team, and understanding the emergence of knowledge sources in communication. Objective 3 of this dissertation addresses this limitation.

> **Objective 3: To examine communication patterns, which includes important communicators and emergent people, and to identify how communication patterns can be represented within a socio-technical congruence model.**

The Value of the Socio-technical Congruence Model

While a number of initial studies have been done on socio-technical congruence and its relationship to software engineering outcomes, such as defect resolution time [50] and code churn [85], the study of socio-technical congruence and socio-technical coordination is still in its infancy. Additional empirical studies that examine socio-technical congruence and its relationship to team performance will contribute to our understanding of the nature of coordination in software teams.

In order to assess the value of the socio-technical congruence model, I use the model to investigate the relationship between socio-technical congruence and team performance. In addition to examining this relationship, I also study the usefulness of the gap size measurement proposed in Section 3.6.2.

> **Objective 4: To apply the socio-technical congruence model to determine the relationship between socio-technical congruence and the performance of a global software-development team working on coordination-intensive activities.**

3.10 What About Technical Dependencies?

A pertinent question at this point is, "What about the technical dependencies?" The limitations discussed up until this point have been largely social in nature. Undeniably, technical dependencies are extremely important and form a large part in fault prediction and detection models [107, 194, 229, 49]. The importance of dependencies has contributed to a large amount of literature devoted to various mechanisms and applications of dependency analysis, such as component analysis [75, 145], feature extraction [86], topic modelling [171], and the use of design structure matrices [185, 141]. Cataldo et al. [49] specifically examined the effect of dependencies on changes induced by failures, including various conceptualizations of the technical relationships calculation (reproduced in Equation 3.1).

As a consequence, the scope of this dissertation does not include technical dependencies based in software code. However, there are also technical aspects that may be adjusted because they are not specifically referring to code, like requirements or features [70]. In addition, because socio-technical congruence is about alignment, some discussion of the choice of technical dependencies is important. I will address conceptualizations of some technical aspects in this dissertation, but I will not be executing studies specifically to explore technical dependencies in software.

3.11 Summary

The preliminary socio-technical congruence model introduced the core elements of the model, including conceptualizations, techniques, and situation modelling. I have com-

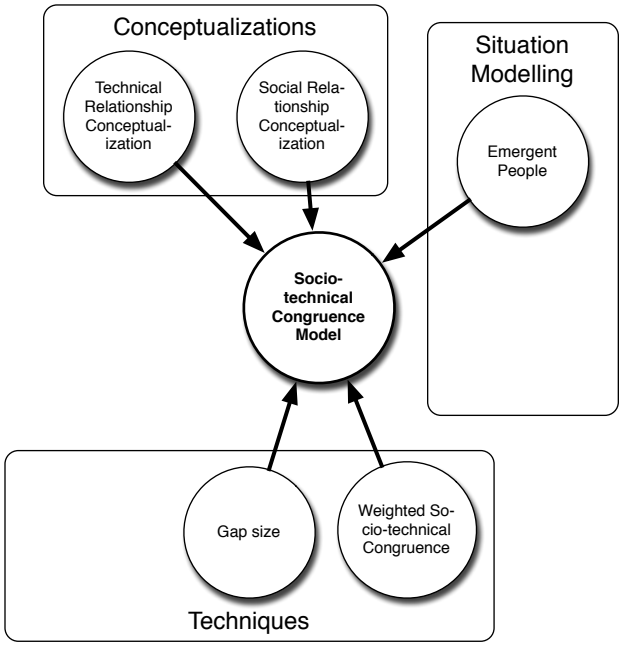


Figure 3.9: The Elements of the Socio-technical Congruence Model Introduced from Related Work

pleted some of these elements based on known related work, reflected in Figure 3.9, and consequently addressed Objective 1.

As I explore the possibilities for incorporating other social aspects into this model and address some of its limitations, I will fill in the remaining elements in the Figure based on findings from Objective 2 (Chapter 4) and Objective 3 (Chapter 5).

Chapter 4

An Empirical Investigation of Awareness in a Global Software Team

A team member with a strong team mental model is able to relate to others and gain an understanding of what occurs within the team. Awareness has been conceptualized as fleeting knowledge about the activities of other team members and is a reflection of the team's mental model. This understanding of the team's capability and awareness of the activities of others has been shown to have an effect on team coordination [89].

This lack of understanding of awareness and its effect on the social and technical work that people do affects the quality of conceptualizations within socio-technical congruence. A study of awareness increases understanding about the aspects of awareness in relation to communication and coordination.

> **Objective 2: To examine awareness and to identify how awareness within a global software-development team can be conceptualized within a socio-technical congruence model.**

This empirical investigation revealed the following findings. I identified that team members communicated using different media with their teammates in parallel, suggesting a need to model multiple relationships in socio-technical congruence. I identified that experienced team members filled gaps, which suggests that conceptualization of social dependencies should include brokerage as a possible relationship. It also reinforced the need to model transitive relationships in the socio-technical congruence model. I identified that not all communication is technical, and that many awareness breakdowns were a result of planning and scoping issues, suggesting that technical entity conceptualizations should be broadened. Each of these findings inspires modifications to the socio-technical congruence model.

In the remainder of this chapter, I describe the Ship project and how I identified the above findings and their relevance to the socio-technical congruence model.

4.1 Research Questions

Research Question 2.1: How do team members maintain awareness of each other's activities in a global software-development team?

To gain an overall understanding of awareness and of the knowledge transmitted throughout the organization, I investigated how a team maintains awareness of each other's activities, and what information they sent to each other. I examined the relationship between awareness and communication, as well as awareness and coordination. By investigating these aspects of communication I identify what aspects of these social interactions can be incorporated into a socio-technical congruence model.

Research Question 2.2: What actions do team members take during an awareness breakdown in a global software-development team?

An awareness breakdown is a situation where a problem occurs because of a lack of knowledge between one or more team members who intended to coordinate with each other. A previous study identified breakdowns occurring as a result of communication overloads [69]; another study identified dependencies between technical components as leading to coordination breakdowns [45]. By identifying problems with coordination, one can highlight the context around the situation and identify what leads up to the problem, what the impact of the problem is, and how to avoid the problem in the future.

4.2 Research Methodology

I used the case study method [227] to examine how a team developing a product called **Ship** coordinated and transmitted awareness information about their tasks to each other. The Ship team is a distributed development team with locations in the United States and Brazil that develops a business-critical product.

A colleague and I spent a period of three months on-site with the Ship team at their Brazil office, which contained developers, testers, and environment coordinators that supported databases and testing environments. My colleague, Sabrina Marczak, was formerly employed by this corporation, but was serving as a researcher conducting her own research during this period. Her involvement allowed me to gain access to the company, its managers, and the developers. In addition, her ability to speak Portuguese built rapport with the team members. She independently conducted her own research [148] on the Ship team in parallel with my own research and data collection.

During the research period, I was permitted to observe their work, sit in on meetings, and listen in on their conference calls. Though the native language in Brazil is Portuguese, the language used within the corporation was English, and the majority of communication in the company was in English especially when remote sites were

involved. I observed working habits at a comfortable distance from the developers, noting when they initiated communication at their computers, used the phone, or had visitors at their desk. I rarely observed one developer for more than half an hour at a time. During this period I took extensive field notes, and after approximately a week of initial observations, I began to ask the team members briefly for context about their work if I needed more details. The employees were very welcoming of the researchers, and were willing to discuss their actions. I had direct contact with every member of the Brazil Ship team, including the Brazil development leader, the test leader, and the contractors.

A questionnaire was administered by my colleague as part of her research investigation. I used demographic data from the questionnaire as a record of the team's experience and role within the project, but otherwise did not use the questionnaire results in this study. The demographics page from the questionnaire appears in Appendix B.

I was able to observe the last week of a maintenance release and the first 6 weeks of an enhancement release. At the end of the observation period, the enhancement release had approximately 10 weeks in its schedule before deployment.

4.2.1 The Ship Project

The Shipping System project, called *Ship* for short, updated and maintained an internal software product used by a multinational corporation. The Ship product was a shipping system that enabled the company to ship its products using different shipping services, or *carriers*. The system operated in many of its manufacturing facilities. The product was approximately seven years old at the time of study. The product was part of a business-critical system, meaning that a failure cost the company a significant amount of revenue.

There were two development sites in this project: United States, and Brazil. The team also communicated with production facilities located in other countries such as Malaysia. The language used among teams was English, but occasionally the Brazil team would revert to Portuguese when speaking only among themselves. The team members worked closely with each other, as well as with individuals outside of their immediate development team. When I came to observe the project, there were 19 team members on Ship. I had personal access to 10 team members in Brazil. Unfortunately, the agreement with the organization that runs the Ship project did not allow me to take software repositories such as source code out of the organization.

I selected the Ship case for the following reasons.

1. It was a global software development team, with development taking place across two locations. The team communicated using electronic media, phone, and face-to-face meetings.

2. It was a commercial project developing a business-critical system. The system was considered integral to the corporation's regular operation. Consequently,

this provided more opportunities to observe behaviour of developers in a potentially high-pressure environment.

3. Ship was a mature team, meaning that the team members have the same mental model. They have worked with each other for at least one year. At the same time, the Ship team has numerous newcomers, allowing us to observe how the team interacts with team members who are still learning the domain and the team.

4. Previous collaboration with the Ship team through my colleague provided me the benefit of increased trust with the team members.

Organization of Shipping System Project

The Ship team was geographically distributed across two sites. The project manager was in the United States; the portfolio manager and the business partner, who were not formally a part of the Ship team, were also in the United States. There were five developers in the United States allocated full-time to Ship, including the senior development leader with seven years experience, and a developer with five years of experience. There were two developers in Brazil allocated full-time to Ship, including the Brazil senior development with three years experience, and two developers in Brazil allocated partially to Ship. In addition, four contractors in a different building in Brazil were allocated in varying amounts to Ship, though Ship is their most important project. There were four testers fully allocated to Ship, located in Brazil. There were two environment co-ordinators, one in Brazil, and one in the United States, who managed the development and test environments. The team members were working together for three years on average, and they had worked together on many projects within the Shipping System portfolio. The team members' roles and experience within the Ship project are listed in Appendix A.

Development/test leaders and senior developers/testers, of which there were several, allocated people to tasks when they receive notification of how many resources they are permitted for the project from upper management. In this respect, leaders and senior team members performed management tasks related to technical project matters.

There were several levels of manager within this organization. Project managers allocated individuals to projects, and from there they are allocated to tasks. These managers also facilitated communication with other parts of the organization, including the business partners, and help define the project's scope and implementation. Product development managers and portfolio managers oversee business operations at a level above projects, and ensured that multiple projects meet the business needs of the organization. In addition to a project manager, there were product managers who governed overall release plans and managed operations on a higher level. Every employee also had a human-resources manager, but I considered these managers to

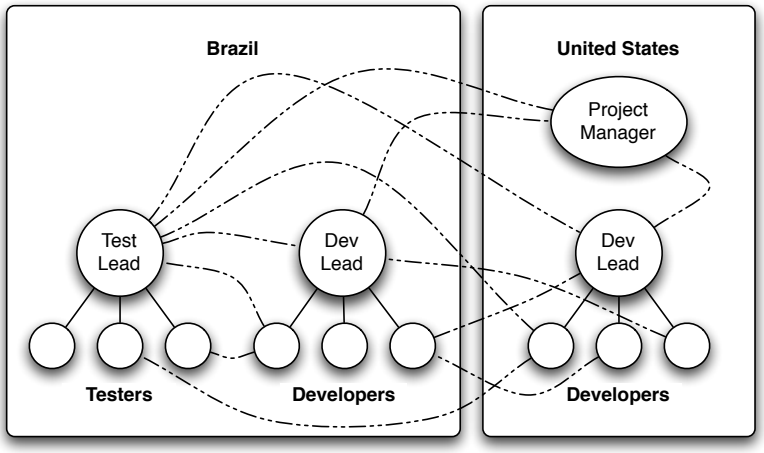


Figure 4.1: Shipping System Communication Structure

be out of scope of the observational study. A representation of Ship's organizational structure is shown in Figure 4.1.

The customer requirements for the software came from internal clients, which the team referred to as business partners, also known as "the business". Business partners are employees of Org, and were liaisons with external business partners from other companies. The business partner communicated primarily with the project manager, lead developer, and senior developer in the United States, as well as the Brazil development leader. Most developers and testers were not encouraged to contact the business partners directly unless it is through a team leader or a senior developer. The business partner was intimately familiar with how the system worked from a top-level perspective, and was able to report in detail on changes that were required in the system.

This system interfaced with a large number of other systems with the company, including order-tracking systems, middleware, and another shipping system, which I will call *Plus*.

The Brazil team was usually four hours ahead of the United States team, though a daylight savings time shift in Brazil in November saw their clocks move forward, increasing the difference to five hours for a period during the project. Consequently, the tendency was that the Brazil team came into work later than the United States team, and usually stayed later to allow for more overlap between the team's working hours.

During our observational period, there were actually two projects occurring simultaneously. The team was finalizing one release and at the same time planning for an upcoming release. I will refer to the first project as *Foxtrot* and the second project as *Tango*. In addition, the gathered data also includes references to a project before Foxtrot that I cannot report details on.

That said, because both Foxtrot and Tango were related to the same product, in this dissertation I concentrate primarily on the interactions between team members

regardless of the actual project. Some team members changed roles from one project to another, which I will discuss below, but the core team members remained the same.

The Foxtrot Project

This project was a technical enhancements project, which means that it was a maintenance release designed to extend features to the Ship product to support its current clients. This project was in development between approximately May and October.

The Tango Project

This project was also a technical enhancements project, and the primary objective of this project was to add software support for additional carriers. In addition, some web-based elements of the system were to be updated. This project was in development between approximately September, and was still ongoing at the end of our data collection period in January. My studies therefore focus primarily on the team makeup involving Tango. The overlapping period between the end of Foxtrot and the beginning of Tango was approximately two months.

I followed up with the team after the project deployment and found that the project deployed on time, and on budget. A number of requirements were moved to a later date because they were determined as high-risk, but the team did not indicate that moving these requirements affected the project significantly.

Support Assignments

In addition to both projects, some team members were partially allocated to perform support work for any issues that were found in the live product. Developers were assigned to support on a rotational basis, and were expected to be on call to rectify any critical defects that were discovered during the operation of the system.

The developers were generally expected to work on support in parallel with other technical enhancement projects, though it was generally understood that support issues took priority over future development.

Changes in Roles Between Projects

There were shifts in various project roles between the two projects, and some people allocated to work on Foxtrot were not allocated to work in Tango. I do not have a complete listing of every affected role but the ones that I am aware of are in Figure 4.1.

Software-Engineering Practices

The team members in Ship used face-to-face interaction, instant messenger, phone, and E-mail extensively; the latter three are used with remote team members. The

Role	Foxtrot	Tango
Senior development leader	Mick	Mark
Development leader (U.S.)	Mark	Ann
Test leader (Ship)	Nancy	Lilly

Table 4.1: Reassigned Roles in Ship project

Brazil development leader often meets with the test lead and the environment coordinator face to face. The culture of the shipping system project encourages communication among team members, even across geographical boundaries. A team member is encouraged to contact any other team member for support.

In a discussion with employees in the company, I learned that the employees take a training course regarding how best to engage in written communication with others. This training included basic English skills and encouragement of practices such as not modifying subject lines and keeping each email to a single topic.

As the company was very large, there were multiple development teams that build and maintain various pieces of software. From what I could observe, no universal software process was used throughout the company, and each team arranged their work practices differently.

4.2.2 Data Description and Collection

Observations and Field Notes I kept field notes in a notebook regarding the team member's activities. I made particular note of their interactions with others. After each day, my colleague and I compared notes and discussed observations made that day to ensure our view of the project and the team were consistent. In addition, I recorded information in a digital log at the end of each work day.

Interviews I conducted 14 interviews with the Ship developers. I interviewed 2 United States developers, the test lead's mentor, and 5 different Brazilian team members. I held multiple interviews with the senior developer (4) and the test leader (2). Interviews were tape-recorded with the permission of the interviewees.

The initial interviews with each team member were semi-structured, with the topic of inquiry being "How are you made aware about changes in the project?" I targeted awareness issues by asking the team members in interviews for instances in which "an unexpected change caused problems" within the team. Based on their verbal description of the problem, I examined supporting documentation and observed the team at work. I also observed the team for any signs of awareness breakdowns, and based on my observations, consulted with team members about issues afterward.

Email messages I collected the inboxes from five Brazilian team members: the senior developer, the test leader, a developer, and two contractors performing development tasks. After arranging emails into threads by subject, I had a total of

5971 unique messages in 1330 threads. I used the *WhatLanguage* Ruby module [58] to automatically detect the language, and manually verified the results. 4163 messages were identified automatically as English, 1449 as Portuguese, and 359 messages could not be tagged with a language. These 359 messages could not have their body contents extracted; for instance they may have contained only an attachment.

I used email messages to augment the situations that team members described in interviews and from observations. I examined the email threads for instances of when team members informed others of their actions and engaged in coordination. Thus, I identified when a person in the thread requested information from others or asked others to execute action, and described the context around such a situation.

I included excerpts of email messages in this chapter. Quotations from email messages were selected because they were considered as representative of email communication patterns observed within the team. Because the emails were collected from developers and testers, the email is biased toward engineering matters. Proprietary information was replaced for non-disclosure reasons, and spelling and grammatical mistakes were corrected to preserve anonymity.

4.2.3 Human Research Ethics Board Approval

The University of Victoria requires that research involving human subjects be approved by is Human Research Ethics Board. The primary concerns of the University regarding human research are *participant recruitment* and *the effect of the study on the participants*. Three deliverables are required as part of the approval process. The first deliverable is an application form which requests information about the research goal and what effects the research has on the organization and its people. The second deliverable is an Invitation to Participate letter that invites a person to take part in the research. The third deliverable is a consent form that the participant signs in order to participate in the research study.

Because we were recruiting within a collaborative industrial environment, we recruited participants by talking with both the managers and employees in the organization. It is important to achieve "manager buy-in", that is, to convince management that the study is valuable for the company and its employees. In my experience with industrial software organizations, employees will not participate unless given permission to do so by management. Thus, the recruitment letter indicated that management had provided permission to the team members to participate, and that their relationship would not be affected regardless of whether they decided to participate in the research or not. Though the human research ethics board may be concerned about managers coercing their employees to participate, the researcher should inform the board that manager involvement actually makes employees more willing to participate. However, the researcher must also ensure that management does not unfairly treat employees who decline to participate in the research study.

Recruitment methods must be described in the form. The participants must be informed of how their information was made known to the researchers. In the

Ship research, this research was acquired from managers, and consequently had to be declared in the Invitation to Participate letter. Finally, the fact that one of the researchers was formerly employed by the company had to be disclosed, and her past involvement as a researcher needed to be identified and that participants should not feel pressured to participate because of her involvement.

As part of the ethics approval process, the effects of participation must be disclosed to the participants. The research needed to be conducted in such a way that it would not put anyone's jobs at risk. Because this study was an observational study, there was minimal risk of harm to the study's participants, and the largest negative effect was that employees needed to use time for interviews with researchers. Privacy was also a risk, which was also handled through anonymization. Data used for this analysis was anonymized and identifying information was removed.

Information sources used in the study must be disclosed to the ethics board in advance. If additional sources of data are to be used, the board must be informed and an adjustment to the form should be submitted. In the Ship study, data collection included observations, interviews, work diaries, and email. Because email was considered as property of the company, explicit permission with the company that the Ship team belonged to, as well as the three contracting companies that the company worked with, had to be acquired. In my study, this was arranged through the company's directors that managed Brazilian development operations.

Further information about the certificate of approval can be found in Appendix C. Due to confidentiality reasons, the entire approval form can not be reproduced in this dissertation.

4.3 Observations of Awareness in Ship

Observations of the Ship team highlighted a number of awareness and communication mechanisms that developers used in their environment. The team members frequently used email, instant messenger, phone, and face-to-face discussions at each others' desks. The team also had Microsoft Sharepoint as well as a network shared drive for sharing documents. I discuss findings of awareness in a socio-technical environment, then identify how these empirical results should be incorporated into a socio-technical congruence model.

4.3.1 Awareness Through Physical Proximity

Distance is often associated with awareness: the closer that two people or two groups are, the more aware they are of each other's activities [85, 72]. The proximity and configuration of the office is known to affect coordination in an organization. Allen [15] reported that offices that there is a dramatic dropoff in face-to-face communication between offices that are over 30 metres apart. In the Ship team, there were two geographical teams, but even within the team at Brazil, their seating arrangement affected the mechanisms that they used to communicate.

The Ship team was not seated together. The lead developer Aaron sat with two other developers from Ship in the same four-cubicle block. The environment coordinator, Kevin, sat approximately four cubicles down the hall from Aaron. The lead tester and one other tester were in another section of the building.

The remaining 4 developers and 1 tester, who were all contractors, were in another building across the pedestrian walkway from the main office building. The buildings were not far from each other, but required exiting and re-entering a keycard-secured area. Thus, only a few developers were in close physical proximity to each other. This necessitated presence indicators, such as instant messenger and phone calls, and required the team to coordinate frequently using digital media, including email and Outlook invitations.

4.3.2 Exchanging Awareness Information

Use of Email and Calendar to Synchronize Activities Email was used extensively in this organization to communicate information, and as a way to keep others aware of their current tasks. It was also used for the purpose of exchanging technical information so that the team members can reach a decision on what actions to take on the project. Because the team used Outlook, email was also used as a scheduling and personal time management tool. Use of the email client varied among team members. Aaron, the senior developer, mentioned the usefulness of "Search" as a way to locate information. Lilly, the test leader, used folders, classified by the project, as well as the source of the message, as a way to organize information. Aaron pointed out that he did not feel overwhelmed by email despite the fact that he had numerous unread messages in his inbox—he simply said that reading email was "a part of his work". On the contrary, the test leader Lilly said that she "sometimes" felt overloaded with incoming email.

Team members occasionally sent out notifications to the development mail list to update others about their tasks:

> **Magda:** Team, just so you all are aware, I have been working on:
>
> - [component] DB Changes to support Residential and Commercial
>
> - Same changes in [product]
>
> - [data] in handling label

This reporting of tasks occurred more frequently among contractors than employees. One reason for this was that the contractors were in another building, and thus relied more frequently on broadcasting their tasks to keep others notified.

Local and Remote Team Meetings to Synchronize Activities Meetings were frequently used as a way to synchronize activities between team members. In general, the Ship team had two types of meetings: face-to-face meetings involving the local

Locations involved	Meeting Type	Description
Brazil only	Meeting room	Team would meet in a common meeting area
Brazil only	Teleconference	Local team would meet over the phone, usually for quick status updates
Brazil and U.S.	Teleconference in meeting room	Brazil team would meet in a common meeting area, and calls into a teleconference with the United States team members
Brazil and U.S.	Teleconference only	Brazil team calls in from their own desks and calls into the teleconference with the United States team members

Table 4.2: Meeting Layouts Among Ship Team

team only and teleconference meetings that involved both the United States and Brazil teams. I observed the configuration of the meetings from the Brazil team's perspective. The configurations are summarized in Table 4.3.2.

The entire Brazil team met once a week. In addition, the test teams and the development teams met independently to discuss issues specific to those teams. The teams also met once a week with the United States team members over teleconference . Teleconferencing meeting locations varied; often the team met inside one of the testing labs that was specifically reserved for the Ship project. Sometimes they would instead met in a conference room.

One drawback of teleconference calls is the lack of awareness of who is participating in the call. Due to the lack of knowledge of who is part of a call, someone occasionally asked, "Is [person] present?" Team members participating in teleconference calls often multitasked. They worked on various other aspects of work, though it depended on the context of the work. In this respect, it illustrates that even if awareness information is sent out to recipients, they do not necessarily receive the information, nor do they necessarily use the information that is offered. Not every teleconferencing meeting involved the remote team—quick status calls often involved the local team only at their desks.

Depending on the phase of development, the team met more frequently. For instance, during system integration testing, which is one of the final stages of testing, the Brazil team met using teleconference at their desks with the United States team every day. Depending on the number of issues, such a phone meeting was anywhere from 10 minutes to 60 minutes long. The discussions tended to be updates regarding the team's status, as well as about the status of important defects.

Presence Awareness with Instant Messenger Every member of the team used instant messenger frequently, and communicated with both local team members and remote team members. Despite the fact that the team members were in the same building, on the same floor, they often used instant messenger.

Instant messages were often sent in parallel with other messages. During teleconferencing calls at the desk, team members discussed various topics. It was not unusual to observe a team member speaking with multiple people in instant messenger during a call; in some cases, the instant messenger conversation was with team members who were involved in the same call that both were listening to. This behaviour was reported by Niinimaki and Lassenius [168] based on interviews.

Instant messenger was used commonly as a presence indicator, also consistent with findings by Niinimaki and Lassenius [168]. An instant message was often used before a person would call or walk over to another person's desk. This is likely because a direct phone call was considered more disruptive than an instant messenger. The use of instant messenger to intentionally switch to another communication medium has been reported by Issacs et al. [126]. The distance between the team members was also great enough such that casual visits would have resulted in a team member not being at their desk.

Instant messenger was also used to augment face-to-face communication in order to send files and data. In one instance, a senior developer received a call from a tester. After diagnosing the problem briefly over the phone, the developer decided to head over to the tester's desk. He was using instant messenger to send information to the tester, even though they were both face-to-face.

4.3.3 Technical Discussions in Email

Technical discussions took place in email frequently, but the developers rarely discussed source code. Instead, they frequently asked higher-level questions about how data was used, and what it was used for. Thus, rather than talking about what fragments of code do, or how to write them, most of the questions were about the meaning of the data. An interview with a developer confirmed that the technical issues of the team were centred around databases and interoperability of the system with third-party components.

Team members often contacted each other regarding coordination issues, but rather than coordinating activities such as code check-ins, they co-ordinated the exchange of information with outside teams who had interdependencies with the shipping system. Often, this information was needed for test cases and system integration, as well as SQL scripts. Unlike in some organizations [28], there was no clear indication in Ship if each component in the system was owned by a single person. Many of their communications were about clarifications about requirements, or about assessing the impact of changes on the rest of the system. Take the following email; Aaron and John are from Ship, while Rita and Mal are developers from another team in the corporation.

> **Rita (developer)**: And... I talked to Mitch, the expert on our team, and he is in agreement with me. We need to determine really, where the Ship [component] table resides, and add the Plus queue information to that table.

> Aaron/John/Mal...any of you guys...can you tell me which Ship server houses the Ship [component] table?

Another example of a technical discussion appears below. Jason was a new contractor, with under a year of experience in this project, and initially reported the problem to Aaron, the Brazil development leader. Aaron acted as the primary liaison between the new contractor and the experienced senior development leader. Even though Aaron and Jason were co-located, both were included in the discussion with Mark.

> **Aaron (senior developer) to Mark, Jason:** If we try to make the comm with [component D] optional, then we will have to re-write some components ([component S], for ex.), because these components read the registry directly...
>
> **Mark (senior development leader) to Aaron, Jason:** Not all orders needs to be in [component D]. Why not let the facility host their own manual process. This way we don't have to worry if this facility needs to send to [component D] or not.
>
> **Aaron to Mark, Jason:** I didn't understand this. Do you mean allow the facility to choose, in the GUI, if it is to be sent to [component D] or not?
>
> **Mark to Aaron, Jason:** No... We will configure it. Once it's configured we'll locked the configuration and only us allow to change it.
>
> **Aaron to Mark, Jason:** And will it be client-specific, ship_area-specific, or facility-specific?
>
> The easiest way is facility-specific.
>
> **Mark to Aaron, Jason:** Then facility it is.

Jason was CCed on each message and was aware of the discussion. Though he was the developer who will be implementing the changes, Mark was clearly the one who made the design decisions. This series of email messages occurred over the course of a few hours in the same day—the initial message (not displayed here) was sent one day before this exchange.

4.3.4 Non-expert Coordination

As software development is an extremely specialized design activity, one might expect that much of the communication that occurs within a team involves specialized knowledge. However, I found that many coordination tasks were in fact quite routine. Nakakoji et al. [163] distinguished between *expertise communication* and *coordination communication*. They stated that coordination communication includes activities such as managing code conflicts. My observations confirm the difference

between expertise communication and coordination communication. Many situations did not require expert knowledge.

A number of the coordination efforts undertaken by the team involved scheduling and planning, rather than technical work. The team leaders were particularly involved in scheduling, as well as managers and the environment coordinators.

One particular pattern I observed in email discussions was a team member requesting permission to perform actions or requesting another team member to execute actions on his or her behalf. This organization, due to its business-critical nature, appearred to employ relatively strict access controls and reviews that individuals needed to work around. Developers and testers interacted frequently with owners of other subsystems, such as various databases, and the environment co-ordinators who managed the servers and test environments.

One discussion of a technical solution occurred between the development leaders, and one of the developers contacted the test team to ask for an approval.

> **Perry (developer) to test leaders:** Can one of you please approve the build request for this change. Kevin is waiting for an approval before installing these changes.

An even more extreme case of the developer not having permission to execute an action is shown below.

> **Perry (developer):** Lilly, Can you attach this email to 2629 and close the defect. I don't have permission to attach emails to the defect.

While the first example suggested that some level of approval was required, the second example does not require the expertise of the a test leader to attach an email to a defect and close it.

4.3.5 Awareness Breakdowns

While present on the site, I observed a number of situations that resulted in confusion and problems within the team. Three different situations stood out as particularly interesting cases of an awareness breakdown.

Situation 1: Domain Knowledge Not Shared

A contractor in Ship, Jason, did not receive a document containing domain knowledge related to one of his requirements, and consequently lost an afternoon of work. The requirement was to redesign a shipping label used by a client to meet localization requirements. Because he did not receive all of the information to code requirements, he designed prototype labels to send to the leaders for feedback. The Brazil senior developer had a document from the client that contained clarifications to a number of requirements, but he had forgotten to forward it to the developer when the contractor was assigned the new label requirement.

The document, which described the clients updated label standard, had been sent to every development lead by a business partner in 2005. The senior developer in Brazil, Aaron, received a new email message on the business partner list, which mentioned the label standard, which prompted him to look up the document from past email archives and discuss it with the senior development leader in the United States and the senior developer in the United States.

Just before a weekly team meeting, the senior developer in Brazil was discussing the prototypes that had been provided with the senior developer in the United States when he recalled the document describing the label standard. They both discovered that the contractor did not receive the updated label standard. The contractor found out about this document in the local team meeting between all team members in Brazil.

Situation 2: Late Requirements Clarifications

The test team of Ship, Lilly, did not receive requirement clarifications from the United States project manager on time, despite the fact that the development team received them. The test lead in Brazil recalled a situation when she sent a list of requirements questions from her team to the project manager. She had also CCed to a senior developer in the United States. When the project manager received clarifications, he informed the developers, but forgot to inform the testers.

Ann, the senior developer in the United States who received the original copy of the message realized that the test team had not received these clarifications, and forwarded them in email to the test leader, closing the awareness gap.

Although this situation was resolved, a delay between sending the clarifications to the development team and sending the clarifications to the test team can mean that the test team works on outdated requirements which may lead to confusion especially when the development team and the test team synchronize.

Situation 3: Deadline Not Communicated

A meeting discussing the deadline for the final day of the planning phase did not involve the test team. Lilly was not informed about the exit date for the planning phase that was being discussed among the project manager and some developers. However, an experienced test lead from the United States, Nancy, who was not allocated to the project, but was acting as a mentor for the current test leader in Brazil, was present in a conference call when she noticed that the project manager was mentioning dates that seemed suspicious to her. She spoke with Lilly about the situation, and then confronted the project manager to ensure that he knew he was forgetting the test team in his planning.

The awareness issue was resolved quickly by the mentor and prevented any damage to the project. However, had the test team not been made aware of the planning phase deadline, coordination in the project would have been affected. They may not have

finished their estimations, their task assignments, or their requirements questions before development started.

4.3.6 Implications of Awareness Breakdowns

Effects of a Technical Awareness Breakdown

Jason, from Situation 1 told us in an interview that he was still productive and did not lose time as a consequence of the awareness breakdown.

> **Jason (contractor):** I don't think I wasted much time because I have other tasks to do. So, if I sent an email [asking], "How should this particular detail should be done?" I did not lose time waiting for them, I have other things to do.

The effect of the delay from global software development was reduced because the developer could work on other tasks. Although some time was lost due to a lack of awareness about the requirement, the time may not have been significant because he was able to work on other, more stable requirements while waiting for more information.

Thus, while there is a significant delay when working in global software development [113, 166], parallelizing work may help minimize the effect of this delay. When planning a project, especially in a global software-development project, a manager should consider assigning a number of stable requirements with an unstable requirement to the same developer.

Effects of a Planning Change Awareness Breakdown

The awareness breakdown in Situation 3 illustrated an awareness breakdown that occurred due to a planning change. In this situation, the change did not involve any technical entities that are traditionally modelled in communication, and was instead involving the project schedule. Part of this awareness breakdown was compounded by distance—the test team does not have as high visibility as development because there are no testers from the project located at the United States, where the project manager was. Fortunately, an experienced former tester, Nancy, identified the problem and was able to communicate quickly back to the test team.

A manager involved in planning must be aware of others when making changes to the plan. This emphasizes the importance of keeping experienced team members at each site who know and understand the needs of other team members, and can bridge their gaps if necessary.

Understanding Information Needs of Others

The information needs of developers in Ship parallel the information needs in previous studies such as those by Ko et al. [132]. Developers seek to understand why technical

modules are implemented in a particular way and often inquired about interactions between components as well. While documentation existed, to some extent, the team members depended heavily on co-workers for accurate, up-to-date information.

A developer's immediate task awareness is acquired by attending meetings or by keeping a close watch on incoming email messages. This information is fed back into a team member's team mental model over time, and this person becomes more experienced and familiar not only with his own relationship to the project, but of the information needs of others.

The development of this team mental model leads to the ability of experienced team members to bridge awareness breakdowns for others. Situation 2 and 3 demonstrated cases where an experienced team member bridged awareness gaps for another team member

In the Ship study, the most experienced team members were in the United States. Some of the awareness breakdowns occurred because decisions were made at the United States location without reporting the results of the decisions to the Brazil team. However, an experienced team member in the United States helped to support the test team leader in Brazil. This phenomenon was observed in a similar study of the Ship team [150].

4.3.7 Team Mental Model of Open Knowledge Exchange

I observed a culture within the team that encouraged the team members to seek advice from each other or provide information when it was requested from others. The team members in Brazil were very willing to assist each other with various tasks. This was reflected in their face-to-face interactions as well as observed in email discussions.

The team communicated frequently with the United States team in teleconference calls, and also exchanged visitors on a regular basis to build trust between the teams. The integration in these teams was very close, and in fact a recent reorganization in Brazil saw the group restructured such that the United States team and the Brazil team would be treated as one team by the portfolio managers. This reorganization reinforced the team's operation as a single unit, rather than as two disparate teams.

The team members reinforced teamwork by giving public acknowledgements to others in email when fixing defects, as below:

> **Thomas (environment coordinator):** I want to thank all of the following people for their help and hard work while working through this problem.
>
> [List of five individuals appears here, addressed by first and last name]
>
> Without their contribution we would not have been able to work the problem through and make all of the corrections.
>
> WAY TO GO GUYS!!!!!!

This was followed up with an acknowledgement by the senior development leader, as well as a shift in focus directing the team toward a new task.

> **Mark (senior development leader):** Way to go team...!! Thanks y'all for your help.
>
> [Project] team, Please validate your process once [dev] confirms that it's working,

It appeared that the developer who initially reported the problem not only wished to express his thanks, but also felt the need to ensure that his appreciation was visible to the senior people in the project.

Even when assigning tasks, the senior developers took the time to send encouragement and recognize team members.

> **Mark (senior development leader):** Team, we've been working really hard on this and spending numerous hours over these changes. We're almost there guys.. Please hang in there with me. Let's make this deployment the same as any other deployment we put together. Looking forward for another success on this. Thanks again for all your hard work and dedication..
>
> Yun, you've step up above and beyond in this release. As always..
>
> **Barry (project manager):** I agree folks. Great job to you all. Be careful in your travels over the next few days too.

In interviews, the team members reported satisfaction with their colleagues and, overall, did not see distributed software engineering as being detrimental to their work.

The team members had regular exchanges between the United States and Brazil. A developer, Yun, went to the United States to help with a data migration needed for part of the project. After two failed data migrations, Perry, Yun, and the senior development leader from the United States, Mark, met for coffee and discussed alternatives—after coming up with a solution they deployed it and the data migration was successful. Yun, who had been working with the team for the past one and a half years, readily praised the two teams that worked over distance. He had just returned from his trip in the United States, working closely with the remote team:

> **Yun (developer):** The team, [Brazil], [United States], we have a really great integration among ourselves. We work very close. It doesn't matter most of the time if you are here in [Brazil] or there in [the United States], because it's the same to work with [the United States team] and [contractor company]. It's easy to work with them, and we have a great integration in terms of the team.

Ann, one of the development leaders, had the following to say about working in the team.

> **Ann (senior development leader):** We have a great team. Working with the Brazil guys is very well done by our team. I think that we are the benchmark for [Organization] in that.

This highlighted the trust that the team members have for each other. The Brazil team is relatively mature, having worked with the United States team for about three years, and at this point they were both working comfortably with each other on the project.

4.3.8 Threats to Validity

The limitation is that the interpretations are made primarily by a single observer. To counter this threat, I have included as many supporting references as I could to support my interpretations. I observed the developers for many days over approximately three months, and reported details with respect to the context in which the Ship team worked. In addition, I examined the data for cases that would explicitly contradict my claims. I have also engaged in *peer debriefing* [83], in which I shared the results of the study with my colleague, who was present with me at the time of the Ship study on site. This allows me to triangulate my observations with an independent observer who is able to identify inconsistencies. However, interpretations of the data based on observations should be replicated in other environments.

Another threat to validity is that a single observer cannot be everywhere at once and thus only get a small piece of the whole picture. The discussions the team members engage in are extremely technical in nature. This makes it difficult for an observer to identify the significance of the coordination issues that the team members face. To counteract this, I complement every observed instance of an awareness breakdown with interviews to confirm their effects on the team members. I approached the team members to understand their work and the effects that coordination has on their work, and later read documents and email collected from the study. The length of the study, over many days, also counteracts this threat by increasing the number of situations that I observed.

The observations made based on this empirical investigation should be considered as evidence of potential phenomenon to investigate further, rather than a generalization that should be applied to all of software development. However, the socio-technical congruence model should be able to handle a large variety of situations—that is, if a situation occurs at least once, then the model should have an adaptation to be able to represent the observed phenomenon, even if it is not considered general behaviour.

4.4 Modifications to the Socio-technical Coordination Model

To complete Objective 2, I discussed how the findings from the awareness study inspire some additional requirements for the socio-technical congruence model. I highlight the contributions from the awareness study to the model in Figure 4.2.

I observed that communication often occurs in parallel, and that multiple media was used simultaneously to maintain awareness. Meetings are easily incorporated as social relationships: simply draw a relationship between one team member and every other person who attended the meeting. If a meeting agenda is available then this can be linked to the topics that were discussed. Furthermore, the results of this study indicated that an **experienced team member** was able to bridge awareness gaps between other team members. This is an example of a **brokerage pattern** [150]. Thus, using a broker can lead to the avoidance of awareness problems, even across distance. However, a broker does not simply "appear", rather, brokerage must be enabled. One way to enable this is to ensure that an experienced team member is located at each site, and that the team member understands the information needs of every role on the team. If a broker B is constantly used to connect two people A and C, a special social relationship network called a **transitive social relationship** can model the transitive relationship between A and C. Such a network would indicate that the two people who communicate through the broker are actually connected and are aware of each other even though they do not directly communicate.

The empirical investigation of awareness highlights the importance of representing different types of communication in parallel. Currently, socio-technical congruence models do not handle simultaneous communication. One way to model parallel communication would be to introduce **multi-variable modelling** by using multiple "social relationship" matrices that each represent a type of communication, and to set it such that certain media types satisfy certain technical relationships. Email, for example, is the primary medium used for detailed technical discussions in this situation and thus should be treated differently from a face-to-face meeting that addresses more design-oriented aspects. Another example would be the use of communication in combination with a brokerage relationship: either one may be considered sufficient to bridge a gap. Using multiple variables may allow one to represent how different levels of **communication richness** or **physical proximity** satisfy technical relationships.

The awareness breakdowns observed in the project were primarily a result of planning and scoping, rather than a result of technical changes. This suggests that **technical relationships may need be expanded** to include elements that are not strictly technical in nature, such as higher-level features. If a feature is modified, then not only should individuals who are "assigned" to the feature be notified, but everyone affected by the feature should be informed—for example, all of the members in a certain role.

The three aspects that appear in the model, *extending the technical relationship conceptualization*, *extending the social relationship conceptualization*, and *multi-*

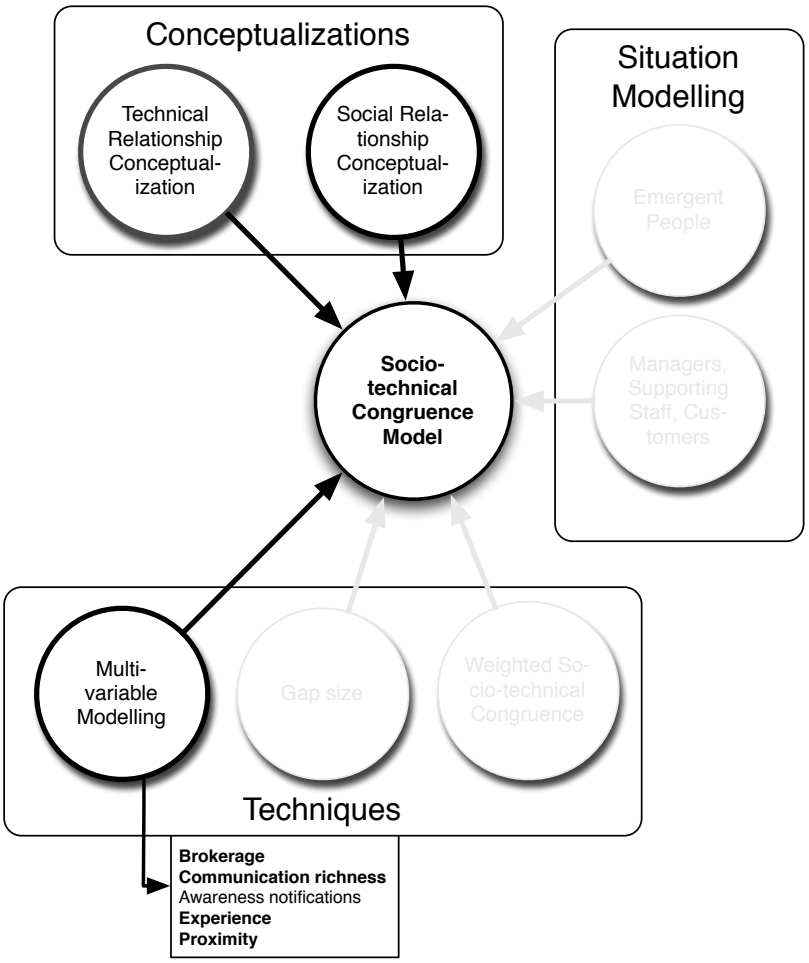


Figure 4.2: Elements From the Awareness Study that Contribute to the Socio-technical Congruence Model

variable modelling, reflect situations that appear in the Ship awareness study. Though these situations were formed specifically from this empirical investigation, it is important for a socio-technical congruence model to be flexible enough to address multiple situations in socio-technical coordination. Thus, these additions increase the ability of the model to be adaptable to a large number of situations.

Chapter 5

An Empirical Investigation of Communication Patterns in a Global Software Team

Communication is an important part of coordination [133, 89] and has an effect on how well teams work together [13]. A socio-technical congruence model is not complete without considering communication.

My objective is to investigate communication patterns in a global software team. Currently, there is little information about *important communicators* and *emergent people* in software development. An important communicator is a person who is highly involved in work-related communication with the team. An emergent person is a person who is included in an online discussion only after an initial set of recipients had been contacted. While current conceptualizations of socio-technical congruence incorporate communication, they do not provide guidelines on how to handle exceptional situations like important communicators, different types of roles, and emergent people. Investigating these situations in socio-technical coordination provides insights into extending the socio-technical congruence model.

> **Objective 3: To examine communication patterns, which includes important communicators and emergent people, and to identify how communication patterns can be represented within a socio-technical congruence model.**

The empirical findings identified that the Ship team development leaders and test leaders communicated extremely frequently, followed by developers. Managers sent messages to large numbers of people. Environment coordinators kept in frequent contact with both developers and testers, more so than managers. Because both managers and environment coordinators were so heavily involved in communication, technical entity conceptualizations should be adjusted to ensure that these people can be included in a socio-technical congruence model. Emergent people were identified to emerge in unexpected situations, in response to requests, during announcements, and

during follow-ups. Emergent repliers were identified as participating to consult with experts, to provide status updates, to report results, and to provide expertise. Thus, when an emergent person appears, that person's link to a technical entity should be identified, if possible. If that person's contribution to the project is significant, then the technical relationships network needs to be updated.

The remainder of the chapter describes the data analysis from Ship that leads me to the above findings. I identify the individuals who are important communicators, and examine why emergent people are involved in discussions.

5.1 Research Questions

Research Question 3.1: What are the characteristics of important communicators within a global software team?

An important communicator is someone who has significance in a software engineering team because of his communication. Communication is commonly acknowledged as important to development work [133], and someone who communicates frequently or is strategically positioned to communicate [38] may be disseminating important information. The objective of this study is to identify common characteristics among these important communicators to determine if they should be incorporated into a socio-technical congruence model.

Research Question 3.2: What situations in email involve of emergent people?

The *emergent people* communication pattern represents an exceptional situation that deserves further attention. During this study I formed a preliminary model describing why emergent people are involved in email threads. A reply made by an emergent person suggests that there was some missing knowledge that required the inclusion of this new person. Studying emergent people has implications for developing effective communication practices in an organization, as well as for expertise-recommender systems, and may lead to a better understanding about why developers provide information to others.

5.2 Research Methodology

I conceptualized communication as the email communication within the Ship team described in Section 4.2.1. The team members confirmed with me that email is the one of the primary methods of communication used within the team; every developer indicated that they relied on email in their work. For them, written communication was important because of their geographic distribution. These team members communicate frequently with team members located in United States and Malaysia, among other locations.

Pattern	Analysis technique	Application
RQ 3.1: Important Communicators	Frequently communicating pairs	Identify roles of people who frequently communicate with each other
	Closeness centrality	Identify team members that contact a wide range of nodes in the network
	Degree centrality	Identify team members who send and receive a large amount of communication
	Eigenvector centrality	Identify team members who talk with other important team members in the network
RQ 3.2: Emergent People	Qualitative analysis	Identify why emergent people are involved
		Identify why emergent repliers reply

Table 5.1: Analyzing Communication Patterns

I conducted two types of analysis to answer the research questions (Table 5.1). I address Research Question 3.1 by applying multiple social network analysis measurements on email communication threads to identify significant participants in the Ship email network (Section 5.4). I address Research Question 3.2 by qualitatively examining email threads in which emergent people are present (Section 5.6).

5.2.1 The Ship Project

For this empirical investigation, I used the Ship project described in Chapter 4, Section 4.2.1.

To provide context for communication patterns, I identified, when possible, the project role of an individual. Many roles were indicated in the email messages themselves, for instance, as job titles in email signatures and in lists of project allocations. The legend for the roles appears in Table 5.2. A list of the people directly associated to Ship appears in Appendix A. To protect the team members' identities, I used fictitious names.

5.2.2 Data Description and Collection

The data source for this empirical investigation was email inboxes from five team members in Brazil. These individuals were the test leader, the development leader, a developer, and two contractors performing development tasks. Prior to processing,

Role	Description
Product/portfolio manager	Manages a product area, which contains multiple projects
Project manager	Manages a project
Manager	A manager (lack of information does not allow me to determine product or project)
System architect	Developer who designs architectures for projects
Senior Development leader	Leads development teams; manages development teams and technical decisions
Development leader	Leads development teams
Developer	Develops as part of a project
Test leader	Leads testing teams
Tester	Tests as part of a project
Business partner	Supplies requirements for projects
Environment coordinator	Sets up and configures environments for project teams
Unknown	(lack of information about this person's role in the project)

Table 5.2: Roles in the Email Social Network

the majority of the emails came from the test leader and the development leader. The email communication occurred among people in the organization that the Ship team members were a part of. The team itself consisted of 21 team members. However, because the team members communicated with people outside of their direct team, the archives involved many members of the organization outside of the Ship team.

The data collected from team members was in Microsoft Outlook PST format. In order to convert this to a mechanism that was usable for social network analysis, I performed the following steps.

- Removed attachments.
- Replaced HTML with plain text.
- Extracted quoted messages into a new message. If a quoted message was identified as embedded in an existing message, it was extracted from the email message and created into its own message. This allowed us to examine email messages that were sent between members of the organization outside of these five team members. Because there was no standard quoting format in the emails, this process was not perfect, and there are a number of edge cases that may have resulted in erroneous parsing.
- Removed non-work-related email. The Brazilian team members very rarely used their email for non-work purposes. Most non-work messages were iden-

tified by discovering email messages that were sent to domains outside of the corporation's domain or the corporation's client's domain.

- Resolved duplicated identities. Because the email archives were in Outlook format and the setting is a corporate network, the identities included names and email addresses. Variations between names were minor; for example "John_Doe@example.com" and "Doe, John". Identities were manually resolved, and double-checked with my colleague who participated in the empirical investigation.

- Grouped messages into threads by subject. Messages with the same subject are considered as part of the same thread. I remove subject line modifiers such as "Re" and "FW". In this organization I found that few members tampered the subject lines. This representation of a thread is a simple, yet sufficient representation of a message thread in this empirical investigation. There were only two exceptions in which this conceptualization did not work: the first situation was when the subject line was left empty, and the second was when the subject line "Tasks" was used. These exceptions were resolved by inspecting the messages manually and combining each message into an appropriate thread.

After arranging emails into threads by subject, I had a total of 5971 unique messages in 1330 threads, where a unique message has a message body, a sender, and a recipients list that is not equal to any other message. I used the *WhatLanguage* Ruby module [58] to automatically detect the language, and manually verified the results. 4163 messages were identified automatically as English, 1449 as Portuguese, and 359 messages could not be tagged with a language. These 359 messages could not have their body contents extracted; for instance they may have contained only an attachment.

The project-related messages did not start until May 1, 2006. I focused my analysis on May 2006 to January 2007, a period of approximately 8 months. I removed 62 messages in 29 threads that were received before May 1, 2006. The various inboxes from the study participants did not necessarily overlap the same time periods—that is, some participants had email archives with a larger time range than others.

The team members rarely removed quoted material from their emails; the entire thread of conversation was included in subsequent replies. Repliers usually placed their reply at the top of a message, above the quoted material. Keeping previous messages intact in a message thread helped both emergent people and initial recipients keep track of the conversation without having to refer to other email messages.

5.3 Analysis of Important Communicators

To answer Research Question 2.1, I used the complete set of email messages for social network analysis, and apply metrics that identify important communicators.

In the email network, each vertex represents a person. An edge v_1, v_2 represents an email that was sent from person v_1 to person v_2. Every edge is directed. Multiple messages between individuals are represented as parallel edges. Loops were allowed. I calculate general statistics of the network and apply social network analysis measurements that are meant to identify important communicators. I apply multiple techniques to ensure that the results are consistent.

Though there were 5971 unique messages in the email network, there were multiple communication edges in the social network because a single message could be broadcast to multiple individuals. Thus, one message sent to multiple recipients, even if it is a unique message, can be "duplicated" due to the nature of email. The social network analysis measurements are applied to the entire network.

The analysis of the email social network was performed using *R* [2] and the *igraph* [63] package.

Centrality

Centrality indicates the relative importance of someone in a social network. In general, a high-centrality individual tends to be "in the middle of things" compared to a low-centrality individual. A centrality value is provided for each vertex in the network.

I used three centrality measures: **closeness centrality**, **degree centrality**, and **eigenvector centrality**. I choose a subset of centrality measurements to use, as some of the existing measurements, such as betweenness centrality, are not appropriate for email networks [35]. I chose not to use betweenness centrality because it calculates the measure based on whether a node is on a large number of shortest-path geodesics. In email networks, no shortest path restrictions exist because email can be addressed to any other member of the network.

Closeness Centrality

Closeness identifies the proportion to which a person in the network is connected directly to others. Closeness indicates that two people communicate without having to go through intermediaries. Someone with high closeness centrality is well-connected to many other people in the network.

Closeness centrality is the average length of the shortest path from this vertex to every other vertex in the network [94]. The closeness centrality C_c for a vertex v_i is calculated as [219]:

$$C_c(v_i) = \Big[\sum_{j=1}^{g} d(v_i, v_j) \Big]^{-1}$$

where $d(v_i, v_j)$ is a distance function calculating the distance between vertices v_i and v_j.

A vertex with a high closeness centrality is considered well-connected to others in the graph [94]. An individual with high closeness centrality has made numerous connections with others and is likely to have many contacts [94].

Degree Centrality

Degree centrality identifies people who have a large volume of communication, and account for parallel duplication of information. Degree centrality indicates the number of incoming or outgoing edges to a node [94]. The node-level degree centrality is simply the degree of the node:

$$C_D(N_i) = d(n_i)$$

The degree centrality represents the number of people who have a direct connection to this actor. Someone who has high degree centrality sends and/or receives a large number of messages, and suggests that this person participates often in communication.

There are three types of degree centrality in directed networks:

In-degree Centrality In-degree centrality is the number of incoming ties to a node [94]. A high out-degree–centrality node receives a large number of emails compared to other nodes in the network.

Out-degree Centrality Out-degree centrality is the number of outgoing ties from a node [94]. A high out-degree–centrality node sends a large number of emails compared to other nodes in the network. Even if a message's content is unique, the message's content is duplicated if it is sent to more than one recipient.

Total Degree Centrality Total degree centrality is the sum of in-degree centrality and out-degree centrality, and is an indication of the total traffic that travels into or out of a node.

Eigenvector Centrality

Eigenvector centrality identifies which network nodes are connected to highly-connected individuals and are thus in a position to learn more about the information moving throughout the network at large. Someone with high eigenvector centrality is more likely to be receiving important information from peers.

A summary of the general procedure is described by Newman [165] and recreated below. Let the centrality of vertex i be denoted by x_i. In order to increase the centrality of a vertex based on the influence of its neighbours, let x_i be proportional to the average of the centralities of i's network neighbours.

$$x_i = \frac{1}{\lambda} \sum_{j=1)}^{N} A_{ij} x_j$$

where λ is a constant. Defining the vector of centralities $\tilde{x} = (x_1, x_2, ...)$, we can rewrite this in matrix form as

$$\lambda\tilde{x} = A \cdot \tilde{x}$$

and thus $\tilde{x}$ is an eigenvector of the adjacency matrix with eigenvalue λ. For more details please consult Newman [165] and the `evcent` documentation in igraph [63].

A large eigenvector centrality indicates that a vertex is connected to nodes who are connected to many other vertices. Thus, while the number of connections still matter, a vertex with a small number of high-quality contacts may outnumber a vertex with a large number of mediocre contacts. In the context of email, a high eigenvector centrality node is connected to nodes that send and receive many messages, and are therefore more likely to receive and be aware of important information in the project.

5.4 Results for Important Communicators

I analyzed 5971 messages sent between 944 unique individuals. Despite the fact that our data set contained of inboxes from only five individuals, our extraction techniques, as well as the frequency at which messages are sent to others in this organization, allowed us to capture a large number of actors in the corporation. Table 5.3 summarizes the characteristics of the email data.

The social network of emails plotted from the perspective of five technical team members appears in Figure 5.1. Most of the network is connected, which is expected. There are two disconnected components: these were emails sent to distribution lists on which one of our participants was on. I was unable to expand the distribution list to identify exactly which team members were on these lists.

Empirical cumulative distribution plots As part of the interpretation of the social network analysis measures, I present empirical cumulative distribution plots that display the distribution of measurements for each vertex in the network. Many social network parameters, such as degree, tend to follow a power-law distribution [135, 12]. I plot the empirical cumulative distribution functions for each type of centrality. These graphs illustrate the distribution of the data. The x-axis indicates the value of the observed measurement, and the y-axis indicates the proportion of observations equal or less than the value at x. For instance, a figure with a very sharp slope at a low value of x (such as in Figure 5.5) indicates that the majority of observations have a low x-value and that very few observations have a high x-value.

5.4.1 Frequent Communicators

I identify how many instances of directed communication occur between each pair of people and identify those pairs that communicate most often. Given the distribution of pairs of people communicating, I took the top 1% most frequent communicating

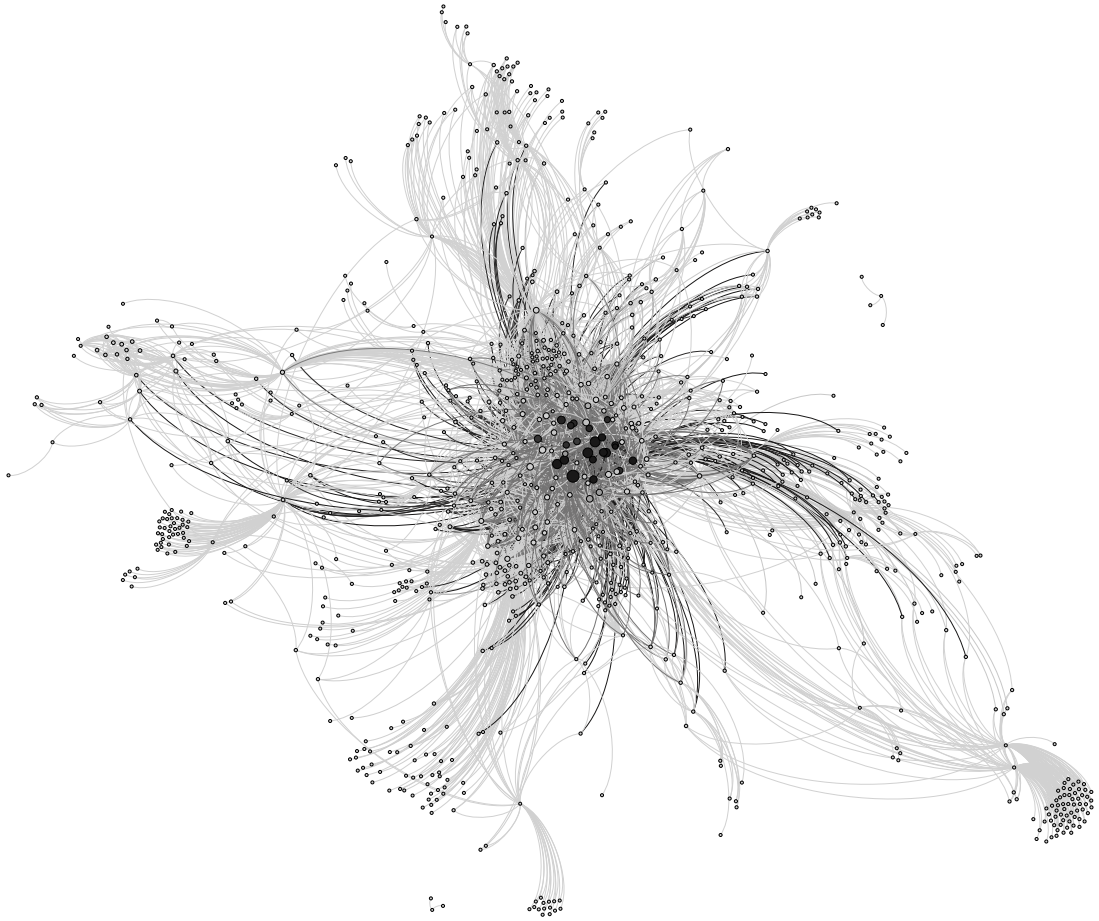


Figure 5.1: Email Network for Ship Based on Five Inboxes from Technical Team Members. The node dize is the Indegree of the node (larger is higher degree) and the greyscale level is the k-core of the node (darker is higher).

pairs. The most frequent communicators communicated at least 87 times (99% percentile). Table 5.4 lists the most frequent communicating pairs in the entire email network. Figure 5.2 shows the social network of frequently-communicating individuals.

The most frequently-communicating pairs of people were the developers communicating with each other, followed by the test leaders communicating with each other. The majority of the frequent 1% communicating pairs were from the Foxtrot and the Tango projects, with the sole exception of Isabelle, who was a senior developer from

Num. of messages	5971
Num. of edges	35856
Num. of unique edges	5348
Num. of nodes	944
Num. of people who sent at least one message	237
Num. of people who received at least one message	924
Num. of threads	1330
Av. num. messages in a thread	4.5
Max num. messages in a thread	80
Av. num. people in a thread	9.3
Max num. people in a thread	96
Av. num. people who sent a message in a thread	2.2
Max num. people who sent a message in a thread	12
Av. num. people who received a message in a thread	8.8
Max num. people who received a message in a thread	95

Table 5.3: Characteristics of Email Data Set

the Plus project. Most of the top communicators were leaders of some kind—either test leaders or development leaders. Only a few developers communicated frequently, but they tended to communicate with the leaders. Of interest was the prominence of the former test leader Nancy, who was previously a test team leader in the project. This person, even when not in a leadership role, still contributed significantly.

The two environment coordinators, Kevin and Thomas, are on this list and communicated frequently with leaders. From the perspective of the development team, the senior developers and test leaders communicated more often with the environment coordinators than they did with the managers in the project. Three managers appear in Table 5.4: these people are Phoebe, the project manager of Foxtrot; Barry, the project manager of Tango; and Sawa, the product manager of the Ship project who oversaw both projects. The managers were relatively low on the frequent communicators list, but one explanation is that the engineering-centred bias of the data did not capture the communication managers had with other managers.

Frequent Communicators within Ship

Figure 5.3 illustrates the communication between the team members that I identified as being in the immediate Ship team for the Tango project. There are 10 Brazilian team members, 9 American team members, and 2 additional American contacts of the team (Sawa and Ari) in this network. With respect to the email network, the core team contained 3.28% of the total actors that I observed. A total of 3657 unique messages were sent within the team, which is 61% of the emails that I observed in this data set. A total of 12324 non-unique email messages (34%), including CCs, were sent by and received within the Ship team. Unlike in the total network that involved

Source	Source Role	Target	Target Role	Connections
Ann	(Development leader)	Aaron	(Development leader)	304
Ann	(Development leader)	Mark	(Senior development leader)	303
Ross	(Test leader)	Lilly	(Test leader)	283
Dan	(Developer)	Aaron	(Development leader)	259
Nancy	(Test leader)	Lilly	(Test leader)	253
Ross	(Test leader)	Nancy	(Test leader)	241
Ann	(Development leader)	Yun	(Developer)	240
Ann	(Development leader)	Perry	(Developer)	225
Lilly	(Test leader)	Nancy	(Test leader)	219
Ann	(Development leader)	Lilly	(Test leader)	191
Nancy	(Test leader)	Ross	(Test leader)	187
Aaron	(Development leader)	Jason	(Developer)	176
Lilly	(Test leader)	Aaron	(Development leader)	174
Aaron	(Development leader)	Dan	(Developer)	172
Lilly	(Test leader)	Adrian	(Tester)	170
Lilly	(Test leader)	Ross	(Test leader)	168
Jason	(Developer)	Aaron	(Development leader)	164
Mark	(Senior development leader)	Ann	(Development leader)	161
Mark	(Senior development leader)	Aaron	(Development leader)	158
Yun	(Developer)	Aaron	(Development leader)	155
Lilly	(Test leader)	Ann	(Development leader)	154
Yun	(Developer)	Perry	(Developer)	153
Jason	(Developer)	Dan	(Developer)	144
Dan	(Developer)	Jason	(Developer)	144
Ross	(Test leader)	Mark	(Senior development leader)	142
Lilly	(Test leader)	Sammy	(Tester)	142
Dan	(Developer)	Yun	(Developer)	137
Mark	(Senior development leader)	Yun	(Developer)	137
Aaron	(Development leader)	Yun	(Developer)	131
Mark	(Senior development leader)	John	(Developer)	130
Ann	(Development leader)	Will	(Developer)	125
Lilly	(Test leader)	Kevin	(Environment coordinator)	124
Perry	(Developer)	Yun	(Developer)	121
Yun	(Developer)	Magda	(Developer)	121
Ross	(Test leader)	Isabelle	(Development leader)	113
Lilly	(Test leader)	Mark	(Senior development leader)	113
Ross	(Test leader)	Phoebe	(Project manager)	112
Lilly	(Test leader)	Josa	(Tester)	105
Lilly	(Test leader)	Yun	(Developer)	104
Ross	(Test leader)	Ann	(Development leader)	103
Aaron	(Development leader)	Ann	(Development leader)	103
Ann	(Development leader)	Barry	(Project manager)	103
Ann	(Development leader)	Thomas	(Environment coordinator)	103
Mark	(Senior development leader)	Perry	(Developer)	103
Ross	(Test leader)	Vig	(Tester)	101
Dan	(Developer)	Carlos	(Unknown)	100
Ross	(Test leader)	Kevin	(Environment coordinator)	100
Ann	(Development leader)	John	(Developer)	99
Kevin	(Environment coordinator)	Lilly	(Test leader)	95
Ann	(Development leader)	Ship Dev ML	(Mailing list)	93
Ross	(Test leader)	Ross	(Test leader)	92
Ann	(Development leader)	Dan	(Developer)	92
Ann	(Development leader)	Nancy	(Test leader)	89
Lilly	(Test leader)	Barry	(Project manager)	87
Ann	(Development leader)	Sawa	(Product manager)	87

Table 5.4: Top 1% communication pairs in Email Network

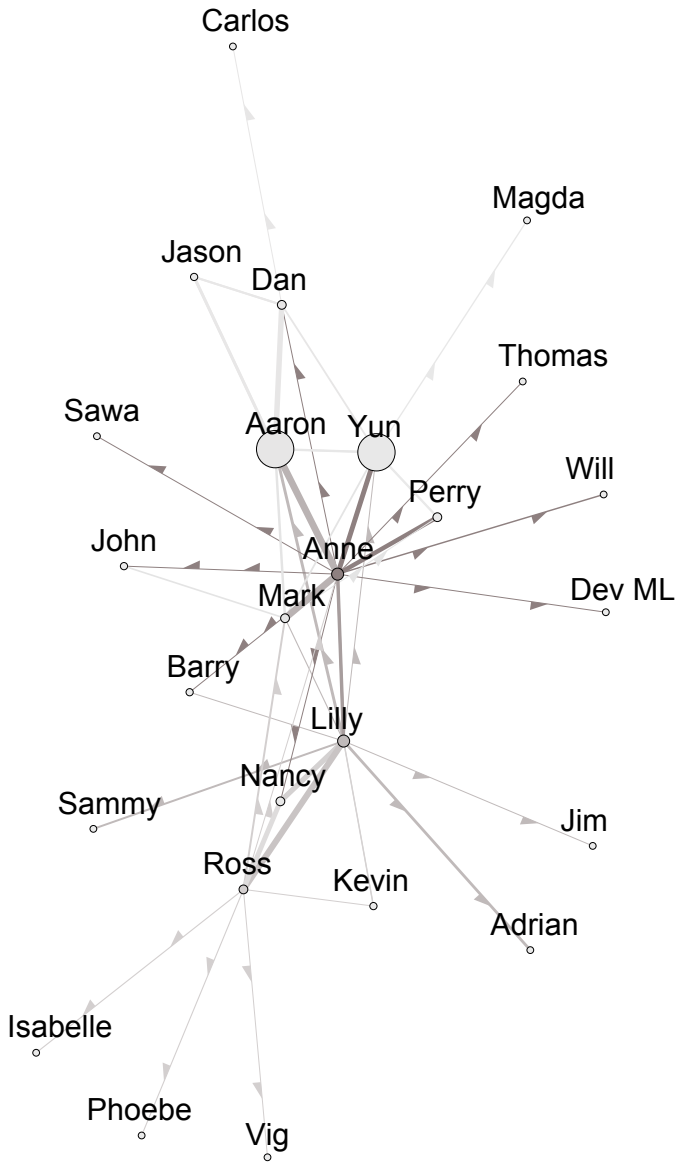


Figure 5.2: Network of High-degree Actors and their Connections. Color is outdegree (darker is higher) and node size is indegree (larger is higher).

every person contacting or who was contacted by the five team members (Figure 5.1), communication within the Ship team is dense; everyone communicates with everyone else.

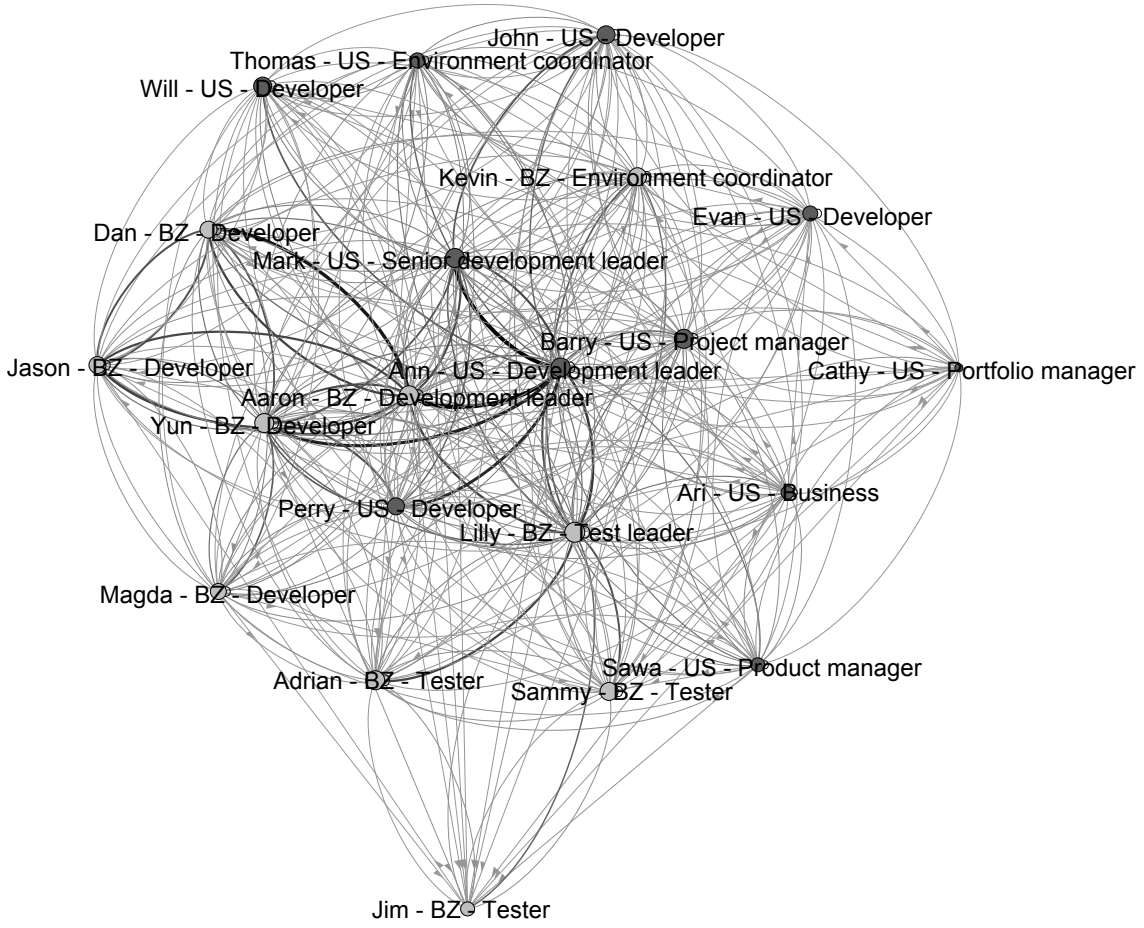


Figure 5.3: Network of Ship Tango Project Team Members Only. The light colours indicate people in Brazil, and the dark colours indicate people in the United States. Node size is indegree (larger is higher). Dark, heavy lines indicate more frequent communication.

5.4.2 Centrality to Identify Important Communicators

Closeness Centrality: Highly-connected Communicators

The closeness numbers indicate the people who are, on average, closest to every other person in the network. Table 5.5 lists the Top 20 members in the network with the highest closeness. Unlike frequently-communicating pairs, closeness measures distance of connections. In this respect, one can see that managers and leaders are in a position to link different people within the organization together.

The majority of these individuals are people who were related to the Ship team members, although there were additional team members such as Nicky (tester from Plus), Cecil (role unknown), Isabelle (developer from Plus), Fred (a product manager), and Ernest (project manager from Plus).

	Name	Role	Closeness
1	Lilly	Test leader	0.111
2	John	Developer	0.110
3	Nancy	Test leader	0.110
4	Mark	Senior development leader	0.110
5	Ann	Development leader	0.110
6	Phoebe	Project manager	0.110
7	Dan	Developer	0.110
8	Nica	Tester	0.109
9	Barry	Project manager	0.109
10	Aaron	Development leader	0.109
11	Ross	Test leader	0.109
12	Cammy	Portfolio Manager	0.109
13	Sawa	Product manager	0.109
14	Ron	Developer	0.109
15	Magda	Developer	0.109
16	Cecil	Manager	0.109
17	Mick	Senior development leader	0.109
18	Yun	Developer	0.108
19	Perry	Developer	0.108
20	Cathy	Project manager	0.108
21	Michelle	Developer	0.108
22	Isabelle	Development leader	0.108
23	Will	Developer	0.108
24	Sammy	Tester	0.108
25	Thomas	Environment coordinator	0.108
26	Trent	Senior development leader	0.108
27	Fred	Product manager	0.108
28	Oz	Unknown	0.108
29	Jason	Developer	0.108
30	Ermi	Project manager	0.107

Table 5.5: Top 30 Closeness Actors in Entire Network

Degree Centrality: High-volume Communicators

Degree indicates the total number of incoming and outgoing messages for each node. This number represents the volume of communication that each person receives or generates.

With respect to email indegree (Table 5.6), the lead tester had the most incoming messages, followed by the Brazilian senior developer; as these two people featured prominently in my email data, this was not surprising. As expected, a very small number of vertices have very high indegrees (Figure 5.5).

	Name	Role	Indegree
1	Lilly	Test leader	2167.00
2	Aaron	Development leader	1992.00
3	Ann	Development leader	1651.00
4	Yun	Developer	1640.00
5	Mark	Senior development leader	1612.00
6	Nancy	Test leader	1234.00
7	Perry	Developer	1143.00
8	Ross	Test leader	1049.00
9	Dan	Developer	893.00
10	John	Developer	823.00
11	Kevin	Environment coordinator	746.00
12	Jason	Developer	675.00
13	Thomas	Environment coordinator	674.00
14	Magda	Developer	632.00
15	Will	Developer	578.00
16	Isabelle	Development leader	540.00
17	Phoebe	Project manager	525.00
18	Sammy	Tester	504.00
19	Ron	Developer	502.00
20	Barry	Project manager	497.00

Table 5.6: Top 20 Indegree Actors in Entire Network

Of the Top 20 degree centrality people, the people that were surprising to see were Kevin and Thomas, the environment coordinators. In this data, which is development-centred rather than management-centred, they each received more messages than the only manager to make the list, Phoebe.

The list of outgoing emails (Table 5.6) is also similar to the in-degree and total

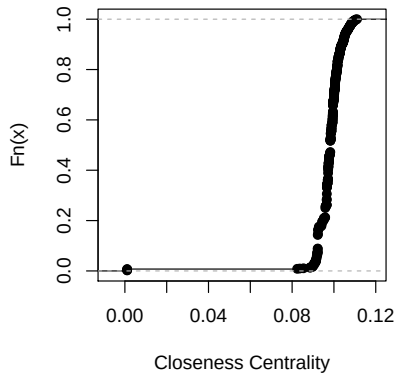


Figure 5.4: Empirical Cumulative Distribution Function Plot of Closeness Centralities

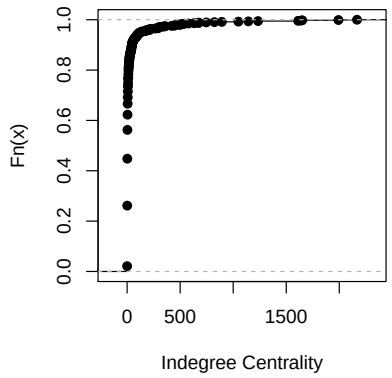


Figure 5.5: Empirical Cumulative Distribution Function Plot of Indegree Centralities

	Name	Role	Outdegree
1	Ann	Development leader	3455.00
2	Lilly	Test leader	3148.00
3	Ross	Test leader	2835.00
4	Mark	Senior development leader	2037.00
5	Nancy	Test leader	1831.00
6	Mick	Senior development leader	1378.00
7	Aaron	Development leader	1317.00
8	Dan	Developer	1278.00
9	Yun	Developer	1217.00
10	Sawa	Product manager	998.00
11	Thomas	Environment coordinator	904.00
12	Barry	Project manager	901.00
13	Cammy	Portfolio Manager	838.00
14	Kevin	Environment coordinator	752.00
15	Jason	Developer	616.00
16	Ben	Unknown	611.00
17	Ari	Business	585.00
18	John	Developer	483.00
19	Phoebe	Project manager	424.00
20	Perry	Developer	402.00

Table 5.7: Top 20 Outdegree Actors in Entire Network

degree centralities. However, the managers ranks highly on the outdegree centrality tables compared to the indegree centrality tables. The business representative also has a prominent outdegree centrality position. Cammy, the portfolio manager of the Plus project, is 13th ranked for outdegree centrality but is not in the Top 20 for indegree centrality, suggesting that managers tended to send emails to large numbers

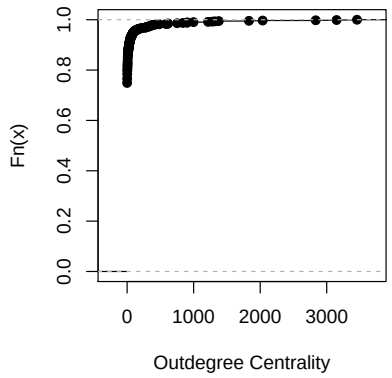


Figure 5.6: Empirical Cumulative Distribution Function Plot of Outdegree Centralities

	Name	Role	Total.degree
1	Lilly	Test leader	5315.00
2	Ann	Development leader	5106.00
3	Ross	Test leader	3884.00
4	Mark	Senior development leader	3649.00
5	Aaron	Development leader	3309.00
6	Nancy	Test leader	3065.00
7	Yun	Developer	2857.00
8	Dan	Developer	2171.00
9	Mick	Senior development leader	1689.00
10	Thomas	Environment coordinator	1578.00
11	Perry	Developer	1545.00
12	Kevin	Environment coordinator	1498.00
13	Sawa	Product manager	1453.00
14	Barry	Project manager	1398.00
15	John	Developer	1306.00
16	Jason	Developer	1291.00
17	Cammy	Portfolio Manager	1011.00
18	Ari	Business	1004.00
19	Magda	Developer	990.00
20	Phoebe	Project manager	949.00

Table 5.8: Top 20 Total Degree Actors in Entire Network

of recipients. Due to limitations of the data set it is not possible to know what number of messages a manager tends to receive.

Our email data indicates that the core members of the Ship team have the highest degree centrality, with the leaders communicating most often. The test leader Lilly, and the development leader Ann both communicated more than the others in the

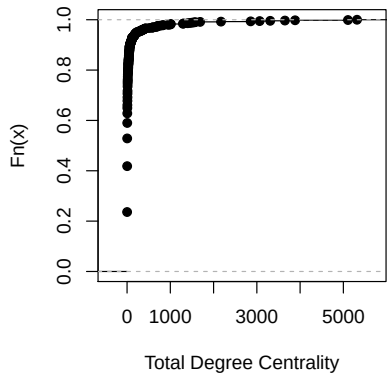


Figure 5.7: Empirical Cumulative Distribution Function Plot of Total Degree Centralities

team, with over 5000 incoming and outgoing messages each (Table 5.8). One unexpected result was that the environment coordinators are very highly ranked, at 10th and 12th place. One of the top-ranked portfolio managers, Cammy, is a member of Plus, not Ship, suggesting that she may have been a point of contact to Ship.

Overall, the test leader and the development leaders tended to be primary communicators. However, even inexperienced developers communicated frequently. Dan, a developer with under a year of experience in the project, ranked highly in each table. Jason, another new developer, also ranked highly. Developers tended to communicate frequently using email compared to testers, who did not make the Top 20 degree centrality table.

Eigenvector Centrality: Connected To Other Influential Communicators

Eigenvector centrality elevates a particular person if that person is connected to important communicators. This means that this person is "in the loop" and tends to receive secondary information from primary sources.

The top eigenvector centrality actors were the development and test leaders. Two actors, the Brazilian test leader and the development leader of Ship from the United States, had high centrality compared to the other team members. The values for eigenvector centrality dropped very quickly outside of the core Ship team (Figure 5.8). Most of the high-profile team members appear in Table 5.9.

5.5 Analysis of Emergent People

To answer Research Question 2.2, I examined emails from the perspective of the five technical team members whose inboxes I collected. The emergent people analysis, therefore, was focused primarily around development and test-oriented activities. I examined emails written in English only. English messages in this data set made up 4105 email messages in 1095 message threads.

	Name	Role	Eigenvector.centrality
1	Ann	Development leader	1.000
2	Lilly	Test leader	0.977
3	Aaron	Development leader	0.769
4	Ross	Test leader	0.766
5	Mark	Senior development leader	0.761
6	Yun	Developer	0.651
7	Nancy	Test leader	0.622
8	Dan	Developer	0.481
9	Perry	Developer	0.399
10	Kevin	Environment coordinator	0.352
11	Jason	Developer	0.335
12	Thomas	Environment coordinator	0.334
13	Barry	Project manager	0.275
14	John	Developer	0.259
15	Sawa	Product manager	0.247
16	Magda	Developer	0.236
17	Mick	Senior development leader	0.233
18	Ari	Business	0.222
19	Isabelle	Development leader	0.204
20	Phoebe	Project manager	0.194

Table 5.9: Top 20 Eigenvector Centrality Actors in Entire Network

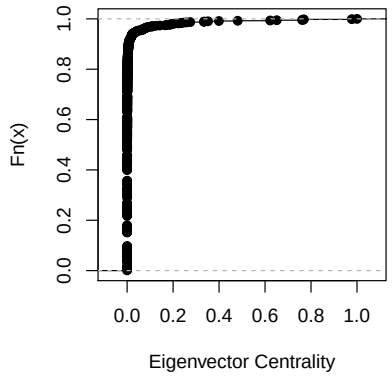


Figure 5.8: Empirical Cumulative Distribution Function Plot of Total Eigenvector Centralities

I applied a script written in R [2] on the social network data that identified the initial list of recipients, and then identified if subsequent messages in the thread added recipients to the list. If any recipients were added, the thread was considered as containing emergent people. If any of these new recipients sent an email message, then the recipients were marked as emergent repliers.

On average, an email thread had approximately two senders and 10 unique involved people in total, though one thread had as many as twelve senders and 96 involved people. An email thread contained on average 4.2 messages, and ranged from 1 message to 76 messages. The number of recipients and the number of emails in a thread follow a power law distribution.

I selected the threads that contained the largest number of emergent incidents, where an emergent people replied to a message in a thread. I read these messages and used thematic qualitative coding to examine the common themes of discussion among the individuals, paying particular attention to when a person emerged in discussion, and when an emergent people contributed to the discussion. After coming up with an initial set of codes, I further selected at random additional threads, some of which may have been duplicates of our initial set, and re-coded the data to verify if the codes were in agreement with the data.

5.6 Emergent People

Of 1095 threads collected from five people, 34% contained emergent people. When there was an emergent person, about 5 people emerged during the course of the discussion. The number of messages in a thread containing no emergent people was about 3 messages/thread, but the number of messages in a thread with emergent people was about 8 messages/thread. The threads that involved an emergent person had more than twice the number of messages than threads with no emergent people. A summary of the number of emergent people in threads appears in Table 5.10.

Of threads with emergent people, 59% threads contained an instance where at least one emergent person replied to the discussion. There was an average of one emergent replier in a thread with emergent people. This means in most cases, someone included in a discussion often had something to contribute.

The distribution of the number of emergent people illustrated one outlying case (Figure 5.9). Ninety-five percent of the threads had 14 emergent people or less. Both the number of emergent people and the number of emergent repliers were heavily

Number of threads	1095
Number of threads with emergent people	458
Number of threads with emergent repliers	269
Av. number of unique emergent people per thread	4.6
Av. number of unique emergent repliers per thread	1.8
Av. number of msgs in threads with emergent people	4.5
Av. number of msgs in threads with no emergent people	2.4
Av. number of msgs in threads with emergent repliers	8.4

Table 5.10: Statistics of Emergent People in Threads

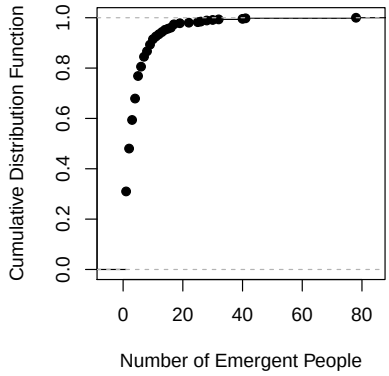


Figure 5.9: Cumulative distribution function of the number of emergent people in email threads

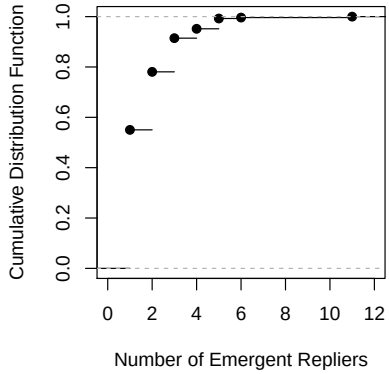


Figure 5.10: Cumulative distribution function of the number of emergent repliers in email threads

skewed to the left (Figures 5.9 and 5.10), and though there were significantly fewer emergent repliers than emergent people, they had similar distributions.

5.6.1 Involving People in Email Threads

An emergent person can be included in the message in one of three ways.

1. The emergent person may be included directly in the CC when the initial recipient writes a reply.

2. The initial recipient may write a reply, and let the initial sender write a new message that includes both the initial recipient and the emergent person in the reply.

3. The initial recipient may write a reply and the initial sender may write a new message but target the emergent recipient only.

Context of emergence	Detailed context
Unexpected Event	Crisis Changing situation Team providing updates
Request	Request and Acknowledge Internal Team Consultation External point of contact
Announcement	General announcement Automated notification forwarding Forgotten team members
Following up	Post-meeting engagement Dormant issue

Table 5.11: Taxonomy of Contexts That Lead to Team Member Emergence

Overwhelmingly, the employees in this organization utilized the first pattern, where the initial recipient replies to the initial sender, and includes the emergent person in a CC. From a process standpoint, this is the fastest way to involve an emergent person. I did not observe the occurrence other two cases in my observations.

Though the pattern of the initial receiver including the emergent recipient in on the message required more effort on the part of the initial recipient, the employees were very willing to engage in this pattern of assistance. In other words, a message participant was willing to take his own time to add someone to the message on behalf of the initial sender.

I could not identify contextual differences of placing someone in the To field rather than in the CC field. Consequently, I treated both fields identically.

5.6.2 Situations That Involve Emergent People

I summarize the contexts in relationship to emergent people in Table 5.11.

There are certain contexts that lead to the involvement of emergent people in the discussion. Initially, these messages are sent to a number of people, but are sent to new recipients.

Unexpected Event

An unexpected, external situation affects the emergence of team members.

The Crisis As the system was a business-critical system that was crucial to the operation of the business, it was often subjected to pressures that were outside of the organization's control, such as infrastructure upgrades, manufacturing stoppages, and

client requests. These unexpected events often affected large numbers of individuals, both within the team that I studied and outside of the team. These unexpected events were *crisis situations* that had to be handled quickly. Often, a manager or senior team member was included in the initial message, and the rest of the technical team emerged as they were notified of the crisis. In many cases, the leaders requested information about how to properly plan for and implement the changes, or delegated the task to an individual on the team.

One example of such a situation was when a major operating system patch was released. These patches were viewed as crucial and a corporate-wide message indicated that they were to be deployed during the weekend, less than a week after the initial notification. In this situation, the test team members were the emergent people. The development leader notified the test leader midway through the conversation, who had to come up with a test strategy of the system once the patches were put into place.

Another example occurred when a development leader was unable to locate a development server. The machine was not responding to pings, and was not in the physical location where it should have been. This issue was a crisis because the server was to be used for user-acceptance testing with a customer who was coming on-site. The emergent people were the development team members; the senior development leader emailed the team to try to locate the machine, and later asked who was using which servers to see which machines could be reallocated.

In crisis situations, the message threads tended to be very long. Many emergent people were notified because crisis situations tend to "snowball", meaning that new recipients were continually added. However, these threads did not involve many emergent repliers, suggesting that even though there were many people aware of the crisis, very few people were actually able to provide information to solve it.

In one case, a solution was discovered after a software upgrade in the environment caused unexpected behaviour in one of the components. After this solution was deployed, the team needed a tester to go through the test scenarios to verify that everything was working again.

Externally-influenced Situation In one situation, a specific development server could not be located in the server room. This required that the team shift resources to new servers. As part of this work, the team leader contacted the development team, who were initially not involved in the problem with the missing computer:

> **Ann (senior developer):** Please reply to all with any dev servers that you are currently working with. We can talk about these later today at the dev team meeting.

The change in situation, due to the server not being locatable, required that the team leader get a status update from the team.

Another situation involved a problem with production servers that had to be rebooted on very short notice. The system architect of the team wrote a message to

the system administrators, as well as the Ship team to inform them of the ongoing situation:

> **John (developer):** The prod [Ship] DB servers listed here are our new Production [environment] that has not been implemented yet so we will need see any issues from these. Once they go into production it would be nice to get this kind of information on a regular basis. Would it be possible to have this report generated weekly and forwarded to all of the [Ship] team?

Even though the entire team, with the exception of the system architect and a development leader were emergent, none of the emergent team members replied, meaning that the message was sent purely for informative purposes.

Some of the situations were not exactly a crisis, but were "unexpected events" outside of the team's control. The participants in the thread would react based on new information provided by communicators involved in the thread. This would often necessitate an action that would be executed by someone not currently in the discussion.

Team Providing Updates—"Peeling the Onion" In this situation, the team members uncover information about an issue as messages are sent from person to person. A message that is initially relevant to only a few people is bounced around, especially from the manager to the senior developer, to the developers. A development manager eventually may consult with his team about a specific issue, which leads to the discovery that technical data may be missing or inconsistent.

Often consulting with the team involved trimming the other recipients so that only the people involved in the technical aspects are included until an answer is agreed upon. When the answer was finalized, often the test team was called upon to test the changes to ensure their correctness, and reported updated information back up to leaders and managers.

Explicit Request

An emergent person often emerged as a result of an explicit request from one of the involved parties in the current discussion as a consequence of what was discussed in the thread.

A developer may explicitly involve a third party during a discussion to request that the developer investigate a specific issue. Even though this emergent person was not involved initially, something that occurred during the discussion triggered one of the participants to request that this person perform a task. Related to this is expertise-seeking: a developer contacts another one to ask for assistance with a technical issue. If the initial recipient is unable to answer, then he will usually refer the sender to someone who does.

In this project, developers and testers do not always seek knowledge from others, but often requested others to execute actions that did not require expert thinking. For instance, the test leader often required that the environment co-ordinators made changes to the test or production environments that the test leader did not have permission to make. Because of the critical nature of this project, access to various components are limited. The environment co-ordinators were often able to make the necessary changes in a timely fashion, but by forcing the team to go through these experts, the environment co-ordinators are fully aware of every change that occurs in the environment. These observations are consistent with the non-expert coordination instances I reported in Section 4.3.4. Other requests I observed included software installations and execution of test scripts.

Request-and-Acknowledge An emergent person can be included because a person was explicitly requested to perform a task or provide expertise. The person requesting the action will often CC the message to the target before sending the message.

This situation will arise when someone seeks expertise, or when someone delegates an assignment in response to an evolving situation.

In this pattern, an individual seeks contextual information from another individual or small group of people. The number of emergent people is small in this pattern because the person who requests information is doing so through someone that he or she is comfortable in contacting.

The information sought after is usually in regards to context, or about the status of the developers. Despite the use of the phrase "expertise seeking" in software engineering to generally refer to contact between developers, most of the requests are not requests for highly-technical information.

In addition, the use of the phrase,"seeks help" often comes up. The developers are looking for help but usually not help that is related to programming.

A sender may ask a recipient to suggest a third party to consult with. This form of expertise-seeking from those that you know well is established in existing literature. (e.g. Nakakoji et al. [163], Erhlich et al. [85]). This pattern is also observed when asking for points of contact for other teams.

During an evolving situation, a team member involved in the discussion may decide that a specific action should be undertaken. This team member will write the request and CC it to the person who should execute the action in addition to the other recipients of the message. The emergent person will reply, usually with an acknowledgement or with the results of the action.

Another trend is one where a problem is discovered, and a technical leader requests the teams to look into the problem. The team that is tasked to investigate the problem moves its specific technical discussions to his own technical team, before reporting back with the results.

For instance, take this question by the test team leader.

Lilly (test leader): Is there a new schedule for [database] deployment?

> We didn't complete database upgrade and [component] test and we'll need to schedule a new set of test to be executed before we can deploy it.

This message was sent to a small number of people, including the project manager Barry, the test team mentor, Nancy, and the development leader in Brazil, Aaron. The reply eventually came from Mark, the senior development leader from the United States, suggesting that Barry forwarded the message to Mark.

> **Mark (senior development leader):** Team,
>
> There shouldn't be anything in the way of the Test team testing [database]. All nodes should be up for [database]. The only valid test outstanding you mentioned below is the [component]. Nothing else is consider in this release. If there's any [tool] then it will be tested with the [Tango] release. Please let me know if you have any questions or concerns.
>
> Please include [John] in all communication related to [database] release.
>
> [Additional list of concerns]

This message was received by the team members that Lilly originally contacted, in addition to John and Mark. These team members were all team leaders.

In many cases, a person is emergent because a decision is made among leaders and the emergent person is asked to execute the decision. This is a case of isolating information within a small group of experts related to the problem. Take for example the following exchange.

> **Hal (developer) to Perry (developer):** The problem is that the testers are not finishing the orders assigned to [P] build, then the delete the order. The delete process does not clear the pallet build locations properly. The best practice would be for the testers to complete orders that are assigned to [P] build.
>
> Otherwise the P build locations that are locked need to be updated.
>
> $$SQLfragment$$
>
> And commit of course.
>
> Hal
>
> **Perry (developer) to Kevin (environment coordinator), Lilly (test leader):** Kevin/Lilly, Here are Hal's comments.
>
> Kevin, can you run the script provided by Hal and that should fix the problem.
>
> Let me know if you have any questions.
>
> Thanks
>
> Perry

This example illustrates coordination between Hal and Perry on an issue and agreeing on a decision. Perry then sent the decision to Kevin, the environment coordinator, to execute the script, and copied Lilly to keep her aware. This situation also illustrates another instance of non-expert coordination (mentioned in Section 4.3.4) between Perry and Kevin.

This type of selective filtering of recipients occurred often within the Ship team. Activities such as scheduling and planning was usually kept within the leaders and was not usually sent to the rest of the team members. In addition, there were many issues that were role-specific and were not forwarded to others until internal decisions are made. The testers and developers tended to do this with various technical issues; in addition, there was also occasional division within the geographic locations, such that the Brazil team would agree on a decision before presenting it to the United States.

Consulting with the Internal Team Managers and technical leaders often served as brokers between teams. This occurred because someone from another team requested information or an action, and the manager included the people on his team because he knew who was relevant to the discussion.

This may have also occurred when a manager caught wind of changes that affected his team; in another case when the manager learned of changes in a requirement, he requested a member of his team to implement the changes.

Consulting an External Point of Contact A point of contact is a person who acts as a representative for another team. This person is available for others to contact if an issue that can be handled by that team arises. While this person may be a manager, this is not usually the case.

In my observations of email, the point of contact was not the emergent person. Rather, it was the people that belonged to the point of contact's team who were emergent, because the point of contact was able to refer the initial sender's request to someone with the ability to solve the issue or provide the necessary information.

One such exchange occurred through the product manager, who contacts a specific individual for a response. In this case, the product manager acts as the point of contact for the Ship development team.

> **Ann (senior developer) to Sawa (portfolio manager) and a number of developers:** Sawa,
>
> The [specification] specifies that invoice value [Text] should be for [carrier1], [carrier2], [carrier3], and [carrier4]. Do all of the other carriers remain as they are today? [carrier5], [carrier6], [carrier7], [carrier8], etc. These all have [Text] printed on their shipping labels (in my sample book).
>
> Thanks,
>
> **Sawa (portfolio manager) to Donny (role unknown) and the original recipients, plus a number of managers:** Hi Donny....

Can you respond to Ann's question?

At the beginning of one of the projects, a list of point of contacts for each team was sent to the team leaders. This list may have been used as a reference by the team leaders when determining who to contact for expertise. However, even this list could not be considered a definitive source of information—there was discussion about how some of the points of contact were not confirmed at the time the message was sent. This concern came up because some of the teams were in between projects, and the people were being reassigned to different roles.

Announcements

Emergent communicators were often brought up when an announcement affecting the project were made. Such announcements are usually wide-reaching, and sent by an upper-level manager. Often, these messages were messages of congratulations, but occasionally contained pertinent project scheduling information.

General announcements Because general announcements were meant to address a large number of individuals, the developers in the project did not hesitate to forward the messages to individuals that they identified as not being on the recipients list. Such bulk mailings are similar to the "snowball effect" observed by Aranda and Venolia [18]. In one important situation, the new project manager sent a message that ordered the postponement of a database upgrade that was to happen this weekend. A developer who had been working for the project for approximately three years noticed that a number of contractors who were working on this upgrade were missing from the recipients list—this developer not only forwarded this message to the missing contractors, but also emailed the manager and informed him to notify the contractors about related issues in the future.

When announcements are deployed, a manager may forget to include someone whose work is affected by the announcement. In these cases, I observed that experienced developers forwarded the message to these forgotten individuals, usually as soon as they received the message. The developer, in one case, also notified the manager that these individuals were not included as recipients and that future correspondence should include these people.

A similar pattern occurred when a manager realized that people were forgotten in an initial email message like a meeting invite, and re-sent the message to the additional recipients.

This indicates that email recipients in this organization are very cognizant of the recipients, and are able to identify when someone is left off of the mailing list. Previous literature [128] indicated that the probability of a reply was reduced if there were a larger number of recipients on the list. Our observation appears to support this result: if all of the necessary individuals are included in the notification then there would be no need to notify the manager that these individuals were missing.

Automated Notification Forwarding In this organization, the managers received regular updates from automated notification systems, particularly of defects discovered in production servers. In these particular instances, the senior development manager determines what information is relevant to his team, and forwards the notification to the development team with a pointer of what is wrong. The team members are emergent because they are not in the initial list of recipients.

This behaviour was observed particularly frequently with a senior development manager in Ship.

Presumably, one reason why the defects are not broadcast to everyone by default is due to information overload. These defects were usually not the same kinds of bugs that one might see during software development, but were critical errors that impede the ability of the products to function. Consequently, it was up to the managers to triage these defects and allocate resources accordingly to repair them.

Following-up

Emergent communicators were often identified as a consequence of following up on an event, which was usually a meeting. The initial message, usually automatically-generated, was sent to a number of individuals.

Meeting Follow-up I observed in a number of cases that the message sent in reply to the meeting invite after the meeting was finished included a number of individuals who were not invited to the meeting.

These emergent people were involved for a number of reasons. First, they may have been invited to the meeting through word of mouth, not email. Second, the people who were included in the reply may be affected by the decisions made at the meeting, and the inclusion of these individuals was a matter of courtesy by the meeting host. In both contexts, follow-up involvement implies that what was initially electronic communication was carried over into other communication media, such as a teleconference or face-to-face meeting.

Reviving a Dormant Issue An issue may be discussed at some length between parties, and suddenly brought up again without any context in the emails. This may suggest that the issue has come up in off-line discussions (meeting, phone, IM) and was important enough to warrant a discussion involving a larger number of people.

5.6.3 Reasons Why an Emergent Person Replies

I examined the contexts surrounding an emergent person's entry into a threaded discussion. Why does an emergent person choose to become involved in the discussion?

In every single one of the threads that involved an emergent person, at least one emergent person responded to the message. On average, of the emergent individuals, approximately half (52%) of them contributed to a discussion in which they were

emergent. This means that people who were included in on a discussion after the fact often had something to contribute back to the discussion, even if they were not initially included.

Even though about five people on average emerged in threads with emergent people, only one person on average sent a reply. This means that while a reply from an emergent person is possible, only a small number of people choose to reply to the email message. I examine why an emergent person chooses to send a reply to the other recipients.

Consulting with Experts

The consulting with experts situation occurs when an emergent person replies to a specific request for expertise or advice.

Initially, the initial sender sends a message with a question to an initial recipient. However, the initial recipient is not able to provide an answer, and thus CCs the message to others for help. The emergent replier consults with the team to request a recommendation on how to proceed. This was not limited to developers; in fact a manager often CCed their technical team for advice. This behaviour is interesting because an emergent person is often presumed to be someone the participants seek information from, and not someone who taps into the discussion to opportunistically gather information.

Status Updates

The status updates situation occurs when an emergent person replies to a request for a team's status.

The initial sender sends a message, usually regarding an unexpected situation, to initial recipients. Because of the problem, the initial recipient then sends this message to a number of emergent people to request a status or a resource update. This applied not only to physical resources, such as computers, but also task assignments within the team. This was most commonly seen during a crisis situation, where an individual sends a message to learn who is using a particular server or who owns a particular piece of code.

Reporting Results

The reporting results situation occurs when an emergent person replies to a request to execute an action.

The initial sender sends a message to a number of initial recipients. Because of the nature of the message, the initial recipient forwards or CCs the message to another person with a request for that person to carry out a task. The emergent replier responds with the results. This occurred frequently due to the limited access issue in the team, where developers and testers do not necessarily have access to make

changes to other components in the testing environment or in the live production environment.

Providing expertise

The providing expertise situation occurs when an emergent person replies to a discussion that involves the initial recipients. Unlike in the consulting with experts situation, there are few emergent people.

The initial sender sends a message to a number of initial recipients discussing an issue. Because the initial recipient does not have an answer, he CCs the message to an emergent person, and the emergent person replies to everyone involved.

An example of this emergent reply occurred in the following exchange.

> **Ben (developer):** Hi Sawa,
>
> I require some help with International requirements for FoxTrot.
>
> [List of requirement-related questions]
>
> **Sawa (product manager):** Hi Ben,
>
> Please review the attached file...it should answer most of your questions.
> As for [question], I will have to ask Norman.

The manager was able to address most of the questions, but for the issue that could be immediately addressed, the manager involved this emergent person. The manager directly included the emergent person in the message, without any kind of introduction or salutation to the issue.

Cover Me! The Cover Me! situation is a specialized version of the "providing expertise" situation. It occurs when an emergent person sends a message to multiple people in order to let others know that he is taking action. In some respects, it is a way to keep others aware of his actions, and allows them to intervene if necessary.

An initial sender, when asking a question, will send the message to an intended recipient, but will CC a manager or a party that is interested in this request in addition to the intended recipient. This lets the manager, as well as the recipient, know who is involved. This also ensures that the recipient knows the importance of the situation, and also keeps the manager aware of the status of the request. If the recipients are unable to take action, then they will often refer the sender to someone who can.

In addition to senders covering themselves, team members often covered for others, especially if they were on vacation or out of office. Many messages have relatively large CC lists; one reason for this would be to ensure that an expert is able to reply even if someone is not actively checking messages. The importance of this is that this organization ensured as often as possible that no critical role was left unfilled, even if other team members were not necessarily aware of someone's absence.

Reasons for Reply	Description
Consulting with experts	An emergent team member consults with the discussants about possible courses of action
Providing status updates	An emergent team member asks for help from the discussants
Reporting results	An emergent team member provides an update or a reply to an earlier request
Providing expertise	An emergent team member supplies expert information about the issue

Table 5.12: Emergent Person Reply Involvement

For instance, one environment co-ordinator Thomas directed his question at the Brazil development leader.

> **Thomas (environment coordinator):** Aaron,
>
> Would the issues I'm seeing on [Server] be related to the [Component] upgrade?

A reply was sent by the Brazilian environment co-ordinator Kevin, who is located at the same site as the development leader Aaron; he redirects the question to Joey, who is not located in Brazil.

> **Kevin (environment coordinator):** Aaron is on vacation. but I think so.. There are several issues yesterday on [component] side, according to Joey.
>
> Joey, any comments?

This pattern of referring people to off-site experts has been observed in previous work [151]. The next day, when addressing the same issue, Thomas and Kevin have another similar situation where they are aware of each other's responsibilities.

> **Thomas (environment coordinator):** Kevin is out so I will bounce some of the [project] Servers and validate the environment.

The team members, in this situation, were able to identify information requests from others and ensure that the appropriate individuals could still coordinate their activities to work around the missing team member.

5.7 Discussion

The investigation of the Ship project provided an in-depth view of how team members on a software project interact with each other through email communication. I examined important communicators and emergent people communication patterns. In this section I discuss some of the findings in the context of socio-technical coordination, and suggest specific techniques that distributed teams can use to mitigate the effects of awareness breakdowns.

5.7.1 Decentralized Knowledge Exchange

Examining only Ship team members in the Tango project revealed a decentralized structure (Figure 5.3). Communication was very dense within this core, and the team members communicated with almost every other team member. The team frequently communicated across boundaries; as shown from our email communication analysis there was no difference in the frequency of contact with remote colleagues compared to local colleagues. This observation reflects the team's mental model of open knowledge exchange, mentioned in Section 4.3.7.

5.7.2 The Contributions to Communication From Support Roles

In this project, it was interesting to discover that the environment coordinators were very involved in project communication, despite the fact that they were not allocated to specific tasks in the project plan like developers and testers are. In fact, the environment coordinators communicated more often with the development team than the managers and the testers. The business partner also was involved in some of the communications, though not as frequently as the core team.

Because this project tended to be database-driven rather than code-driven (Section 4.3.3), this may explain why the environment coordinator is a critical team member. In addition, the environment coordinators worked primarily with the Ship team and did not have assignments to many other projects. This may have led to better integration of the environment coordinator into Ship.

In addition, the managers send many messages compared to other team members. Though they are extensively involved in communication in the same way that support-staff are, and are not necessarily in a central location of the network, they were shown to send messages to large numbers of people. This implies that managers require more support for reaching large subgroups of individuals—perhaps individual teams, or sets of roles—than other roles.

5.7.3 Rapid Communication in Crisis Mode

A situation that often required knowledge exchange in this project were situations that were "crisis situations". These situations were easily identifiable because they included a large number of emergent people and deep message threads. However, these crisis situations usually did not include a large number of emergent repliers. In a crisis, communication is spread to as many relevant recipients as possible on the chance that any one person with the right expertise can step in with a solution.

Previous work by Dabbish et al. [66] and Kargiannis and Vojnovic [128] has shown that the more recipients there are for a message, the less likely a reply to that message will be received. The style of communication during a crisis in Ship appears to be counterintuitive—one would think that it is of utmost importance to receive a reply in such a situation. Additional analysis would need to be done to determine (a) what the probability of receiving a reply is in a crisis situation is and (b) whether the reply received leads to solving the problem, even if the initial message was sent to many recipients.

While a crisis situation is often related to technical issues, like a missing server situation, or a defect in the production system, these are not always recorded in technical repositories. Socio-technical congruence models using repository data would capture the second case, since it goes through a repository, but not the first one.

5.7.4 Locating Emergent Experts

Locating the correct people to contact can delay progress on work. Thirty-five percent of the threads sent included emergent people, and of these threads, 58% included an emergent replier. Threads that included emergent people are longer, on average, compared to threads that did not include emergent people (2.45 vs 7.47). These threads were longer because portions of the discussion were devoted to letting everyone know that an additional person was included in the discussion. If a message is addressed properly to the right people from the start, then there is less coordination setup required and more knowledge can be exchanged.

Automated knowledge management systems have been built for the purpose of locating experts in order to reduce communication delays. Examples of these systems are STeP_IN [226], Expertise Browser [160], Expertise Recommender [155], Related Contributors [173], and EEL [157]. Unfortunately, there is little evidence that recommender systems have gained wide adoption in software development. In this particular organization, the team members were content with going to other team members for information. Recommender systems would help in a subset of cases that I observed as involving emergent people, and specifically, a recommender system that is able to locate appropriate points of contact may be more useful than a system that refers individual developers or testers. With respect to extremely technical information, a point of contact in a gatekeeper role may be aware of who has knowledge in his team, but an individual developer, if asked a specific question, not only might not know the answer, but might not know who to refer the questioner to.

A team member's communication network within an organization is also very extensive—one person simply can not keep track of every person in his network. Much of the communication in the Ship team involved people who were outside of the Ship team, but have interdependencies with the Ship team. These people included business analysts, points of contact outside of the company, operations staff from the company's shipping facility, and technical experts from other projects such as Plus. The amount of interconnectivity the Ship team had with people outside of the immediate project team was impressive. The fact that over nine-hundred names could be extracted from the inboxes of five team members with incoming emails from the past six months shows how far email can reach within the organization. Some simple contact management tools may benefit team members in a socio-technical environment immensely. Most team members do not need references to frequently-reached contacts—they know their team members. What they do need is a way to contact important people that are infrequently spoken to, but are necessary to move project tasks along.

5.7.5 Threats to Validity

The main limitation of this study is that the email data is not a complete sample and was based on inboxes from five team members in development. I was only able to gather a small cross-section of data from the Brazil team members. This means that the email messages tend to be development and testing centred, and likely favour interaction among the Brazil team members. At the same time, gathering an entire snapshot of the company's email messages would have been extremely difficult and subject to a number of ethnical and logistical problems. As a consequence, the quantitative analysis associated with this data collection method is only valid for a subset of the team. This data set reflects the activities of the development team the best, because the majority of the emails were collected from team members who were developers. However, the test leader had the single largest inbox collected.

Because the email data set was primarily developer and tester-based, managers and other supporting staff are underrepresented in this analysis. Though I could infer that managers, customers, and supporting staff were important communicators, I was unable to determine the content of their discussions. For example, though I found that managers received and reacted to automatic notifications, I cannot conclude whether managers found these useful, or if their position in a network was inflated because of such a system. My limited access to manager inboxes does not allow me to draw conclusions about what information managers receive in this project.

The data is focused on the Ship Brazil team and consequently email from the United States is underrepresented. This means that local email communication within the United States team is not in this data set, nor are communications between managers recorded. As a consequence, I try to limit discussion of these groups to the nature of their outgoing messages rather than drawing conclusions about the communication needs of managers. Even though a substantial amount of data could not be

retrieved, I believe that the insights benefit the study of socio-technical coordination.

The data was in a non-standard format and although reasonable technical effort was expended in ensuring the correctness of the data, there may be instances of duplicated messages, missing messages, timing errors regarding message time stamps (particularly with respect to transitions in daylight savings time), or similar. I do not believe that the interpretation of the data is affected by these errors.

While many of the roles of the primary actors within this data were identified correctly, some of the roles and tasks of members within the network could not be confirmed, making our understanding of the circumstances limited. In addition, because of the small number of individuals examined in detail, the assessment of roles cannot be considered as complete. The insights into the team's communication patterns indicates potential areas for exploration—by identifying phenomenon for exploration, the groundwork can be laid for future studies that involve a larger number of cases.

With respect to the emergent person analysis, the fact that 1449 messages were in Portuguese may affect the reasons that team members emerged. The coding of emergent person situations was done based on sampling from the 4163 English messages. Possible effects would be the existence of more casual and personal interactions between team members in Portuguese messages, or the explanation of complicated tasks that the team members may not have felt comfortable explaining in English. Nonetheless, as the majority of the messages were in English, there is sufficient evidence to establish the preliminary model of how emergent people are involved in conversations.

Care was taken not to use bias when understanding emergent people using qualitative methods. Unfortunately the data was coded by a single individual only. I went back to the data and examined the email with the intention of looking for exceptional cases that did not fit the categories.

5.8 Modifications to the Socio-technical Coordination Model

To complete Objective 3, I discussed how empirically-observed communication patterns inspire a number of necessary guidelines when applying the socio-technical congruence model. The main finding is the incorporation of the emergent person. I highlight the contributions from the email communication patterns study to the framework in Figure 5.11.

This research investigated important communicators in Ship. The takeaway is that **support staff, as well as managers, are important team members whose continued involvement is crucial to the project**. These team members do not have direct technical dependencies, but yet have a large amount of influence on socio-technical coordination—for instance, the environment coordinators are often required to run and install scripts in the Ship project. One way to extend the involvement for these individuals would be to **refine the conceptualization of a technical**

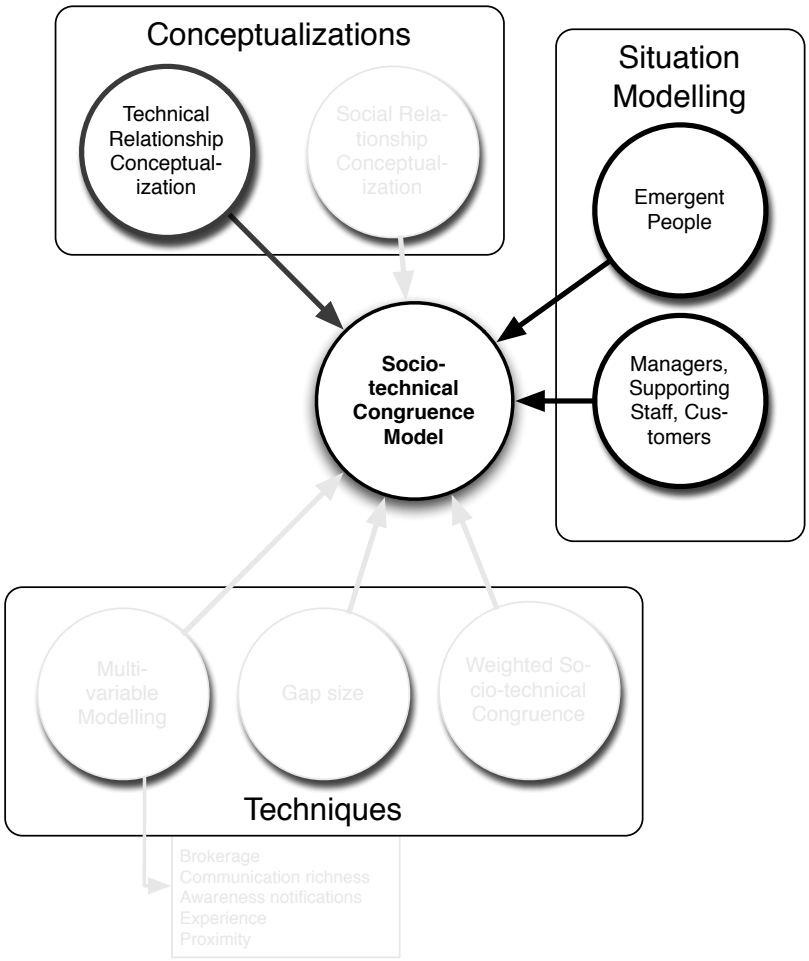


Figure 5.11: Elements from the Email Communication Patterns Study that Contribute to the Socio-technical Congruence Model

entity to include managers. Another way to incorporate all of the staff would be to represent multiple technical entities: for instance, there may be a technical entity for source code, but another one for all code changes that require the operation of a particular database server. The changes that require the database server would each create a technical relationship between the people involved with the change and the appropriate environment coordinator. Thus, similarly to the conclusions in Chapter 4, the concept of a technical relationship can be expanded to include these roles.

The emergent person emerges for a number of reasons. In some cases, such as a crisis, the emergent people are not necessarily involved in the situation and do not contribute back. However, receiving such messages increases the awareness of team members. The Ship team members exchanged information within their own team frequently. This may reinforce existing research regarding familiarity [85], highlighting that people who are familiar with each other are more likely to contact each other for information. Thus, an emergent person appearing in another team member's network

is not detrimental to the project if both are already within the same organizational group. A more important situation is when the emergent person is brought into the conversation through a broker; this connection tends to be more tenuous and there were a few messages where Ship team members were sending point-of-contact information for teams such as Plus. The takeaway is that **certain types of emergent people—the ones that are outside of the immediate team—are surprising** and are possibly important to acknowledge to avoid delays.

The presence of emergent people, as well as support roles, highlights a need for **time-sensitive technical relationships**. If an emergent person is important, then that person's connection to the technical entity should be identified so the technical relationships network can be updated. However, not all team members necessarily need to be included at all times. An obvious example is when an environment coordinator needs to execute a script for a tester only after testing is completed. He can execute it only after the development team has implemented the changes, and the test team has written up the test scripts. A technical relationships network that adjusts to the current process would be valuable.

Chapter 6

Applying the Socio-technical Congruence Model to Study Socio-technical Congruence and its Effect on Team Performance

In this chapter, I applied the socio-technical congruence model developed in Chapter 3 to examine the relationship between socio-technical congruence and the performance of a global software team called IBM Rational Team Concert (RTC). I used this opportunity to apply the gap size measurement from Section 3.6.2. There are only a handful of existing studies about socio-technical congruence [50, 48, 84, 101]. While each of these studies thus far highlighted that good congruence is beneficial to a project, no studies exist that examine in depth the relationship between socio-technical coordination and the performance of a team executing a coordination-intensive activity such as a software build.

> **Objective 4: To apply the socio-technical congruence model to determine the relationship between socio-technical congruence and the performance of a global software-development team working on coordination-intensive activities.**

This study identified that there is a significant relationship between socio-technical congruence and team performance. However, this effect was an interaction effect, meaning that the effect depended on the context in which the study is performed. The gap size and the weighted congruence measurements were also identified as useful conceptualizations by identifying that there are high gap sizes in successful builds, but low gap sizes in failed builds.

The remainder of this chapter describes the empirical study of RTC in detail, including the conceptualizations of socio-technical congruence as well as the empirical findings.

6.1 Research Questions

Research Question 4.1: Does socio-technical congruence have an effect on build success probability?

I conceptualized a coordination-intensive activity as a software build because a build incorporates multiple technical artifacts written by team members. I conceptualized "the performance of a team" as the *probability that a software build succeeds*. A build result, which can be *error* or *OK*, indicates the relative health of the project up to that build. If a regularly-scheduled build succeeds, the team was able to coordinate up until that point. If the build fails, there was a potential problem in coordination. Studying the relationship between socio-technical congruence and the performance of a distributed development team working on a coordination-intensive activity tests the applicability of socio-technical congruence, as well as its limitations.

Previous work in socio-technical congruence [50, 48] has established a precedent that increased socio-technical congruence correlates with increased performance within software teams. Thus, in the RTC project team, I hypothesized the following:

> **Hypothesis 4.1:** Increasing socio-technical congruence correlates to increased team performance.

Research Question 4.2: Does gap size have an effect on build success probability?

Socio-technical congruence has limitations, such as the lack of gap priority, that were discussed in Section 3.3. To address these limitations, I developed an improvement to the existing socio-technical congruence calculation by adding weighted congruence and gap sizes, as described in Section 3.6.2. The improvement consisted of three steps. First, I formalized a technique to identify the location of gaps. Second, I defined a technique to weigh the relationships between dependencies and people in the socio-technical congruence formulas. Third, I devised a technique to measure the size of a **gap** between two people who are technically dependent. Thus, in addition to examining empirically the effects of socio-technical congruence, I examined whether the amount of coordination that occurs between two people should be proportional to the number of dependencies they have.

I applied the gap size measurement to the RTC project. A gap in socio-technical congruence suggests that there is a lack of coordination where coordination should be occurring, and thus could be considered as a negative influence. Ehrlich et al. [84] also found that increased numbers of gaps led to increased code check-in incidents. Following this logic, a large gap size would be considered less desirable than a small gap size or no gap at all. I hypothesized the following.

> **Hypothesis 4.2: Decreasing the gap size correlates to increased team performance.**

6.2 Research Methodology

I conducted an empirical investigation of the IBM® Rational Team Concert® (RTC) software product. To measure socio-technical congruence I applied two different measures: a previously-published congruence approach [50], and a weighted congruence approach that provides details about the size of a gap between two individuals [139]. I also examined RTC's processes and tools to identify any explanations of the relationship between congruence and builds that I find.

6.2.1 The IBM® Rational Team Concert® Project

RTC is a collaborative software development tool built on the Jazz™ architecture. It incorporates the integrated development environment, the source-code control system, the issue-tracking system, and collaborative development tools. The team writes both the client software and the server software. The RTC team self-hosts, meaning that the team uses its own software to develop subsequent versions.

The following qualities made RTC an attractive case in which to study socio-technical coordination:

1. The distributed team. Because distribution makes coordinating work difficult [118, 111], RTC is supposed to enhance communication among project members by providing collaborative facilities [95]. Developers explicitly record communication across sites using the product.

2. The use of an advanced electronic collaboration tool. The RTC developers use their own product to develop new versions. Consequently, the RTC client itself is a socio-technical environment.

3. The iterative process. RTC's iterative process encourages developers to "build early and often". This provides a large number of "checkpoints" against which we can evaluate the team's performance.

4. The rich data repository. RTC saves a large amount of information, including the links between work items, builds, source code, and comments.

Because the RTC team uses the Rational Team Concert software, it is a socio-technical environment that incorporates social influences as well as technical ones. The environment in which they work provides notifications about their own activities, as well as the activities of others. RTC incorporates a developer-centred "dashboard" that reports on units of work called *work items*. As a part of this reporting, a user knows what he is currently assigned, the status of these work items, and changes made to these work items. The user also receives updates to work items that may be related to his current work, as well as work items that he has specifically set to watch. Figure 6.1 shows a portion of "Team Central", the developer-centred dashboard.

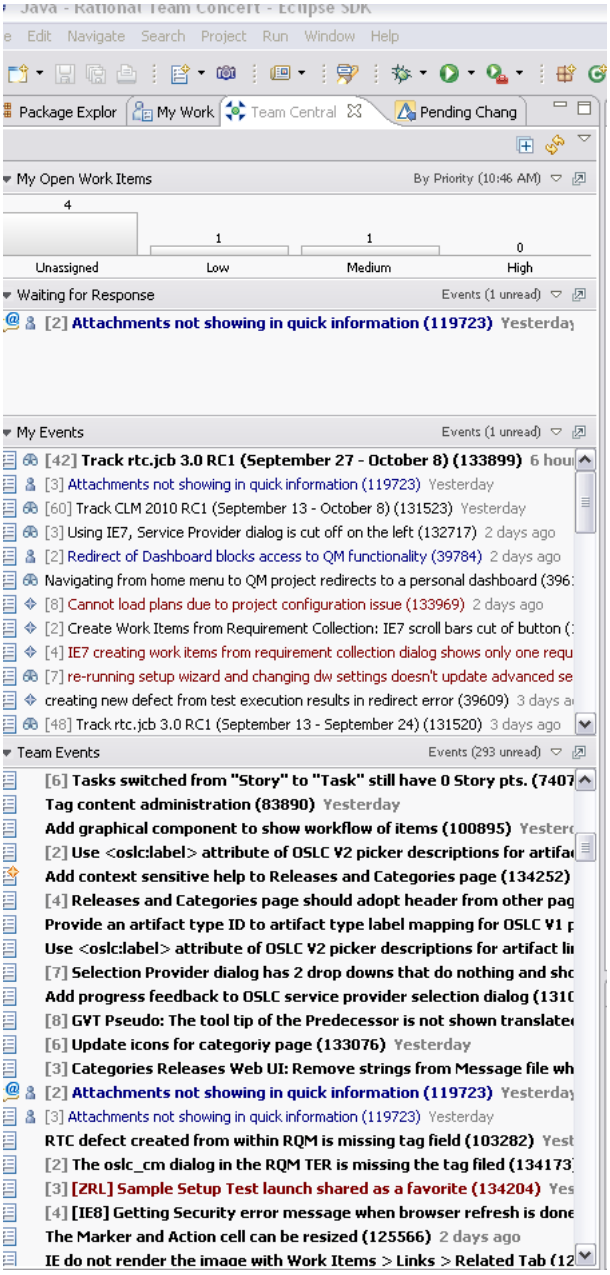


Figure 6.1: RTC Team Central Showing a Developer's Personal Events and Team Events

Organization of RTC Project

At the time of my study, the RTC development team was distributed over 16 different sites located in the United States, Canada, and Europe. Seven sites were active in RTC development and testing. There were 151 active contributors at these locations. Each team, which was not necessarily confined to one geographical location, was responsible for developing a component of RTC. The team sizes ranged from 1 to 20 and had an average of 5.7 members. The number of developers per geographical site ranged from 7 to 24 and was 14.8 on average.

Software-Engineering Practices

The project used the *Eclipse Way* development process [95]. The process defines six-week iteration cycles, which are separated into planning, development and stabilization activities. A project management committee formulates the goals and features for each release at the beginning of the each iteration, and *work item*s represent assignable and traceable tasks for each team. RTC's process encourages frequent building, including continuous, nightly, weekly, and integration builds.

One unique aspect of RTC is its "open-commercial" development model. RTC is a commercial product developed by IBM, but has a publicly-accessible issue-tracking and reports system. This open-commercial model allows a large amount of information to flow into the team from the community.

Individuals communicated through issue-tracking comments, instant messaging, Internet Relay Chat, by phone, and face-to-face. Personal email was discouraged. The team used a mailing list to deliver announcements, such as server maintenance schedules. The team recommended that discussions that did not occur in public channels should be summarized as work item comments so that remote team members are able to track decisions and rationales.

6.2.2 Data Description and Collection

In collaboration with IBM, I acquired a copy of the RTC repository containing data over a one year period. This period of time covered twelve "milestones", varying from 1 week to 4 weeks long, during which the RTC team achieved a number of objectives. The end of the data set coincided with a major milestone in which RTC was to be released for open beta testing.

In addition to quantitative data, I obtained contextual information about the development team through interviews. I also investigated work items, comments, and reports available on the `jazz.net` web site.

The repository I collected from IBM contains build information, work items, comments, change sets, and anonymised author information. I describe the data in more detail below.

Work item. A work item is a description of a unit of work to be done. A work item can be assigned a type of *task*, *enhancement*, or *defect*. I extracted 2008 unique

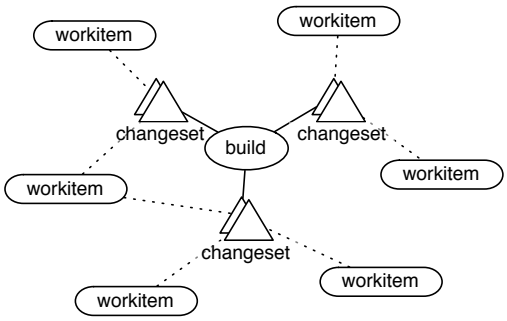


Figure 6.2: Relationship among Builds, Change Sets, Files, and Work Items

work items that were relevant to my study. The same work item may be involved in multiple builds; a total of 9218 non-unique work items were associated with a build across the data set. As my conceptualization focuses on the per-build level, I must consider these non-unique work items.

Build. A build is the process of compiling the software to create a working program. A build outcome can be an *error*, which indicates that there was either a compilation error or a test suite error, *OK*, which indicates no errors, and *Warning*, which indicates that there were warnings returned by the compiler or the test suites.

A build contains the work from one or more work items (Figure 6.2.2), and these work items are not necessarily unique from build to build. The RTC repository does not keep a full record of every build over time, which means that I have more data points for the two most recent milestones compared to earlier in the project.

The RTC repository contained 533 builds, but I considered only builds that contain technical relationships, therefore dropping the number of builds to 214. I removed 17 builds whose result is Warning because the way that the RTC team treats its Warning builds is dependent on the build. Finally, I removed 7 builds whose types could not be identified. The resulting data set contained 191 builds.

There are three types of builds that I investigated in this study: *continuous builds*, *nightly builds*, and *integration builds*. Continuous builds, which were run regularly by a user, incorporated changes from the local development site, plus known stable components from remote sites. Nightly builds also incorporated changes from the local site and stable components from remote sites approximately once a day. Integration builds integrated the latest components from every development site. Integration builds were often expected to break due to their large complexity, but errors in continuous and nightly builds were indicative of potential coordination problems. The data contained 122 continuous builds, 55 nightly builds, and 14 integration builds.

Because I had a data set of builds over time, I included the date of a build as a variable in my analysis. The build date was conceptualized as the number of days after the first build in the data set.

Comment. A comment is written text authored by a developer that is about a particular work item (Figure 6.7(b)). Comments are the primary method of transferring information among developers in RTC. Multiple comments may be attached to a single work item. The data subset contained 9323 comments.

Change set. A change set is a collection of code changes to a number of files (Figure 6.7(a)). When a developer updates the repository, all of the files that were changed together are updated in a single change set. A change set is generated by one author only, and is related to exactly one work item. A single work item may contain multiple change sets, and the same change set may occur in multiple builds. There were 3013 change sets in RTC. The number of change sets per work item ranged from 1 to 246. Ninety-five percent of the work items contained 15 or less change sets, and 42% of the work items contained only one change set.

Source code file. A file contains source code and is included in change sets. Over time, a file may be associated with multiple change sets.

Author. An author is someone who has contributed to RTC. In my conceptualization, I specifically identify an author as meaning someone who has submitted a change set containing modifications to files. Because each change set can be attributed to one author, there are always at least as many change sets as there are authors in a build.

I chose to use build success probability as my response variable for a number of reasons.

First, I decided to use build success probability rather than defect resolution time, which is the amount of time that a defect is open for. Defect resolution time is often not indicative of the actual time that an issue is worked on, and thus is a coarse estimate of the quality of the team. Just because a defect is opened does not mean that a developer is actually working on it. As a case in point, a small number of open issues in the RTC project have been in the repository for over two years, and are simply viewed as not important enough to be addressed in light of more recent, urgent issues. In addition, a defect is often meant to be completed by a single person rather than by a team, suggesting that defect resolution time may be a better indicator of individual performance rather than team performance.

Second, the nature of the RTC project, given its frequent branching and merging capability, allows developers to commit code changes frequently. As such, I do not feel that code changes are an accurate performance measurement that evaluates the team's effectiveness. One perspective may see code changes are good—the developers are committing code frequently and are productive!—but alternately the viewpoint

may be that frequent code changes are bad—the developers are checking in a lot of mistakes and are having to fix them. A more objective measurement is needed.

Thus, build success, in the context of RTC, is a stronger measurement of team performance up until that point in time than defect repair time or code changes. A build usually requires multiple contributors that commit multiple code changes. I also confirmed with an RTC manager of the importance of successful builds—he said, "Build quality is almost code quality". Because there were multiple builds over time in this project, I was able to compile the builds and discover the contribution of congruence to the probability of success.

Distribution of Build Data

Each build involved a number of file authors, change sets, work items, and files. The same author, file, and work item may be involved across multiple builds. I plotted the histograms for each of these entities in Figures 6.3(a), 6.3(b), 6.3(c), and 6.3(d). Most of these distributions, especially the files, were skewed toward the left[1].

Figure 6.3: Distribution of Authors, Change Sets, Work Items, and Files per Build

I plotted the progression of builds over time for unweighted congruence (Figure 6.4) and weighted congruence (Figure 6.5). The data indicated that builds were

[1]This heavily left-skewed distribution is consistent with findings by Zeltyn, et al. [228]

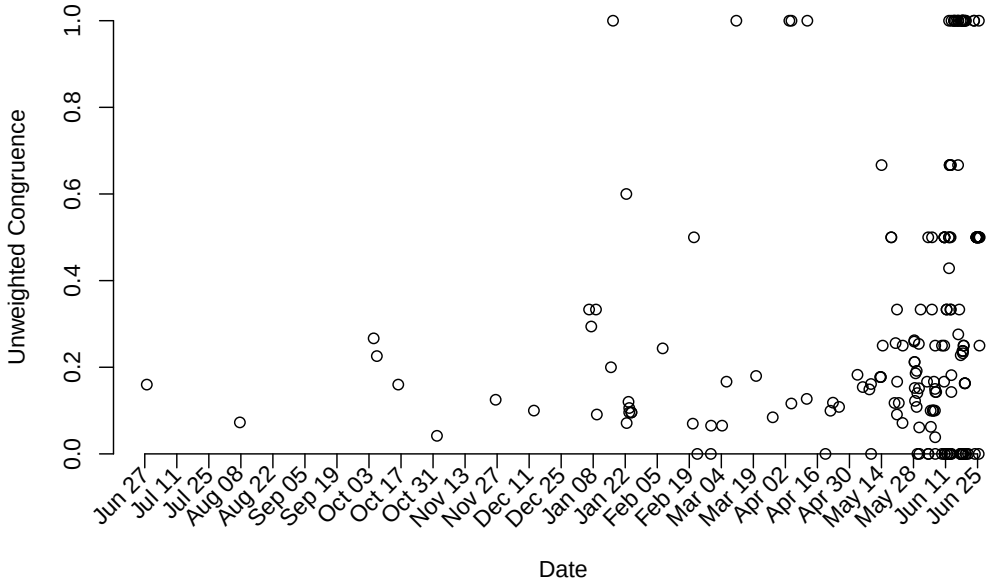


Figure 6.4: Unweighted Congruence Over Time

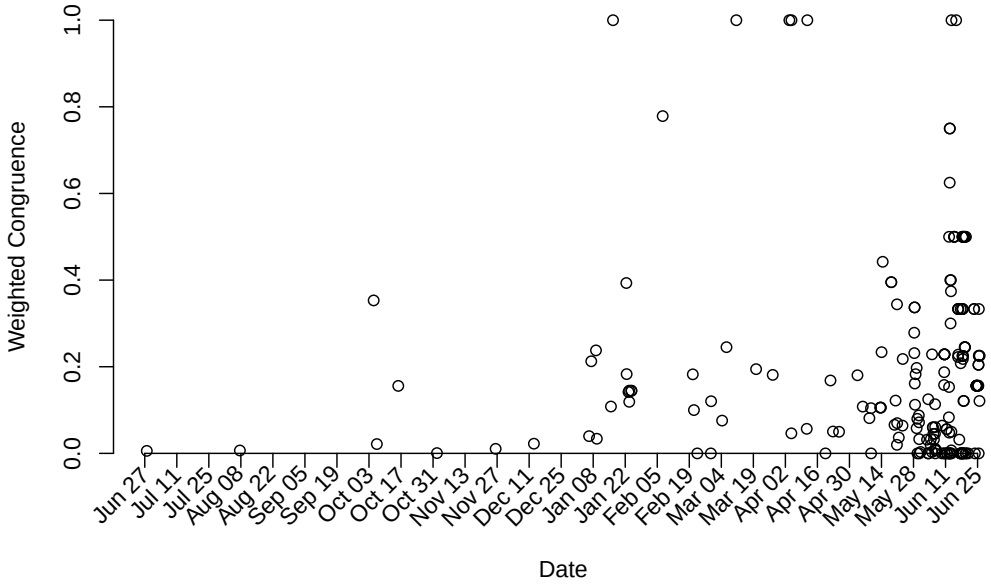


Figure 6.5: Weighted Congruence Over Time

more frequent in the later phases of the project. This is expected, as the release iterations became smaller as the project matured.

6.2.3 Analysis Method

I used the unweighted congruence measure and weighted congruence measure presented in Section 3.6.1 for my analysis. I explain how I conceptualized congruence in this section.

The socio-technical congruence template appears in Figure 6.6. This section follows the template's layout, explaining each relationship.

The template revealed that a number of assumptions had to be made in this project. For example, I assumed that developers coordinate predominantly through work items, and the change of a same file within different work items necessitated communication. A work item is attached to one or more change sets, but because not all work items are necessarily dependent on each other, it is likely that the communication that occurs in this project with this conceptualization is underestimated.

Conceptualizing Technical Relationships

I decided to construct dependencies based on a variation of the files-changed together heuristic because a "files-changed together" heuristic was more reflective of the RTC team's work. Their work was constructed heavily around work items and change sets, and thus it made sense to use a logical approach based on change sets.

I presumed that a technical relationship between two individuals exists if, minimally, these two individuals change the same file within the same build. In RTC, since only one author can be associated with each file check-in, two change sets are involved in the mapping of a technical relationship, one associated with each file author. These two individuals should communicate with each other to ensure that the appropriate file dependencies are properly handled before the software is built (Figure 6.7(a)).

I generated the technical relationships matrix as follows.

1. **For each build, I identified every change set included in the build.** Every change set between the previous build and the current build is considered a part of the current build.

2. **I determined authorship for each file to generate the assignment matrix *A*.** Since each change set had only one author, the author was assigned to every file in that change set. If an author modified the same file in a different change set, then I added an additional edge for each change set in which the author changed that file.

3. **I determined the file dependencies in a build to generate the dependency matrix *D*.** I iterated through every change set in the build. Each file

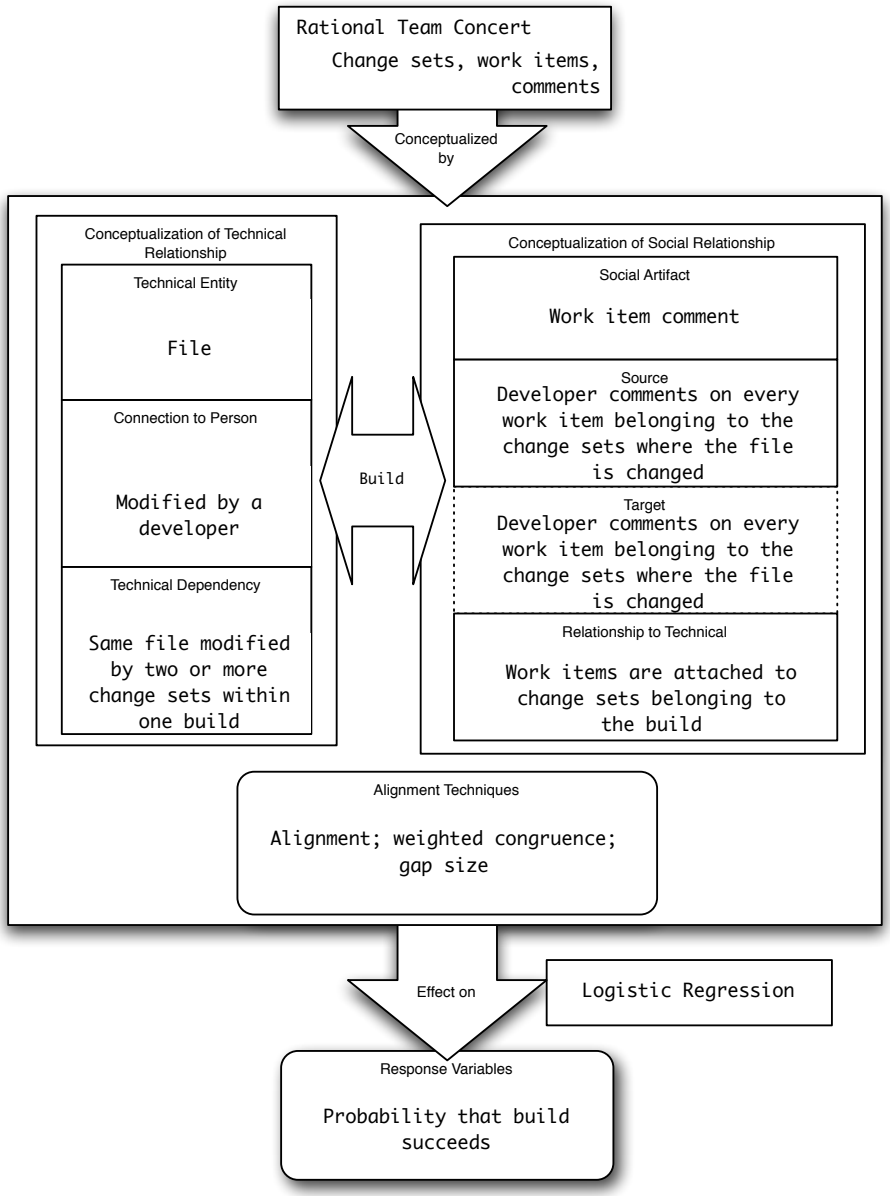
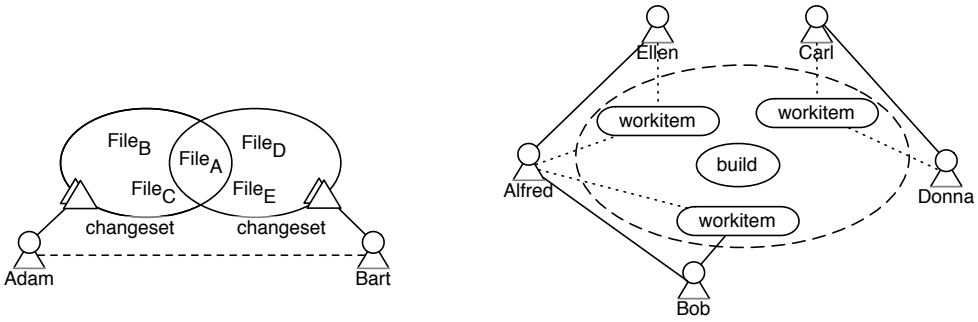


Figure 6.6: The Socio-technical Congruence Template Applied to the Rational Team Concert Study



(a) Constructing a Technical Relationships Instance

(b) Constructing an Social Relationship Instance

Figure 6.7: Conceptualizing Socio-technical Congruence in IBM Rational Team Concert

was dependent on every other file within a change set. For example, if there were two change sets C_1 and C_2 in Build 1, with C_1 containing files A, B, and C, and C_2 containing files A, D, and E, the resulting dependency matrix D would correspond to Figure 6.7(a). In weighted congruence, I added one additional edge for each file pair that existed in each additional change set in that build. Thus, in a Build 2 with change sets C_3={A, B, C} and C_4={A, B}, the dependency matrix entries in D would be $D_{AB} = 2$, $D_{AC} = 1$, and $D_{BC} = 1$.

4. **I calculated the technical relationships as per the technical relationships formula (Section 3.6.1, Formula 3.1)** The diagonals were ignored. When applying weighted congruence, I normalized the technical relationships matrix by taking the highest-ranked edge and dividing every entry in the matrix by that value. This converted each edge to a value between 0 and 1, indicating the relative rank of a technical relationship between two individuals in the matrix.

I used this conceptualization because of the explicit relationships between change sets, builds, and authors in RTC. In previous "files-changed-together" approaches, multiple files checked in together were associated with multiple authors, whereas in RTC, a change set is authored by one author only. My conceptualization of technical relationships represented a situation in which two individuals work on a file that were in two different change sets, but were within the same build. Though the file-level changes may not have necessarily been to the same lines of code, I presumed that multiple changes within a single file were sensitive enough such that each developer working on that file benefitted by being informed of each change within a build's time period.

Conceptualizing Social Relationships

I conceptualized social relationship as communication through comments in the RTC environment. I treated those who commented on a work item as a group of communicators, and assigned weights when individuals commented on multiple work items that others also commented on (Figure 6.7(b)).

Though the RTC commenting system allowed users to post multiple posts to the same work item, I did not count multiple posts in a single work item as weighted communication. Since a work item comment could be read by anyone with access to the work item repository, and there is no threading in the work item comments, I could not assume that there is point-to-point communication from one person directly to another. In addition, comment communication within the same work item did not correspond to weighted technical dependencies that crossed different change sets.

I calculated the social relationship matrix as follows.

1. **I identified the comments from the work items involved in a build.** Each build contained a number of change sets, and each change set had an associated work item with comments. For each work item, I included every comment that was posted after the previous build, but before the current build started.

2. **For each person who commented on a work item, I added a communication edge to every other person who commented on the same work item.** In weighted congruence, I added one additional communication edge for each additional work item in which two individuals comment. I did not count weights for multiple comments in a single work item for the reasons discussed above. Thus, if work item W_1 contained commenters A, B, C and W_2 contains B, C, D, the resulting entries in the social relationship matrix SR would be $SR_{AB} = 1$, $SR_{BC} = 2$, $SR_{CD} = 1$, $SR_{AC} = 1$, and $SR_{CD} = 1$.

3. **When applying weighted congruence, I normalized the communication network by taking the highest-ranked edge and dividing every edge in the network by that value.** This converted each edge to a value between 0 and 1, indicating the relative rank of comment-based communication between two individuals in the network.

This conceptualization of weighted communication represented the number of people who post a comment to the same work item.

I considered incorporating the RTC development mailing list into communication, but inspection of the mailing list revealed that the developers did not talk about technical issues and that the development list was primarily used for general announcements.

Unfortunately, due to the distribution of the 151 developers in the RTC team, as well as the one-year time period across which my study is conducted, I was unable to collect communication data such as instant messenger or face-to-face communication

that I could associate to individual builds. My concern with this missing data led us to inquire about the use of other forms of media among the team. I learned that although unofficial communication does happen between developers, the RTC team culture requires that the content of unrecorded and private communications be recreated as work item comments for consumption by remote project participants. I still expected that socio-technical congruence will be lower than I might expect from such a team as a consequence.

Conceptualizing Gap Size

The gap size in RTC represents a distance between the technical relationships and social relationship. Since the two measurements were not directly comparable to each other, I normalized both measures to identify a relative "rank" of technical relationships and social relationship. For example, if there is a technical relationship in a build that has many dependencies, intuitively I would want a larger amount of communication around that build rather than on a build with few technical relationships. The gap size calculation identifies this mismatch.

I used the weighted congruence measure to compute the gap size. For each build, I calculated the gap size between each pair with a technical relationship using the lack-of-coordination matrix explained in Section 3.6.3. Because of the normalization, each gap size is a value from 1 to -1, where 1 is a large gap with little or no communication, 0 is full congruence, and -1 is a gap where there is more communication than necessary to fulfill the technical relationship. I take the mean of these gap sizes and use this as a variable in a logistic regression model. I also consider an alternative calculation of gap size where I do not punish extra communication and set all values of gap size less than 0 to 0.

Statistical Analysis Methods

To answer Research Question 4.1, I conceptualized congruence as described in Section 6.2.3 and test Hypothesis 4.1 by using logistic regression.

Logistic regression is ideal to test the relationship between multiple variables and a binary outcome, which in this study is a build result being either "OK" or "Error". The presence of many data entities in this project means that I must consider confounding variables in addition to the socio-technical congruence when determining its effects on the probability of build success. Informally, logistic regression identifies the amount of "influence" that a variable has in the probability that a build will be successful.

I show the relationship between a variable and the build success probability by plotting the y-axis as the probability. I use probability because it is more intuitive than odds ratios or logistic functions. If there is a relationship between a variable and the probability of build success, then as the variable's value increases, the probability also increases. In the probability figures, the solid line is the expected value, and the dashed lines indicate the 95% confidence intervals.

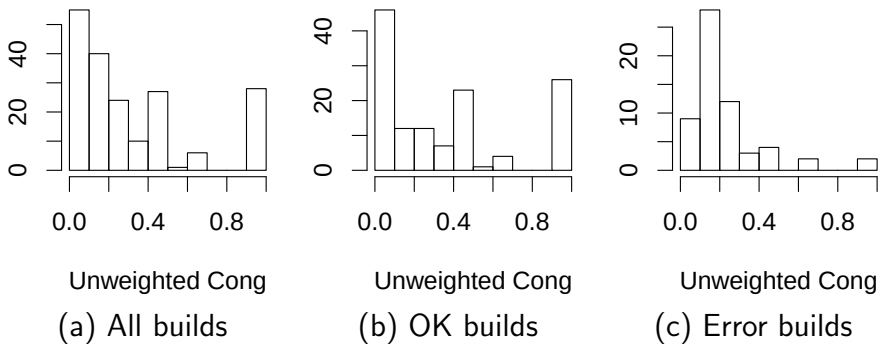


Figure 6.8: Distribution of Unweighted Congruence Values

I ran two different logistic regression models: one using weighted congruence, and one using unweighted congruence. I included the following variables: number of files per build, number of authors contributing to the build, number of files in the build, number of work items per build, the congruence, the build type, and the date of the build. I centre and scale each numeric variable. Because I was concerned about possible interactions affecting my results, I included first-order interaction effects and used backward stepwise elimination to remove variables to keep AIC (Akaike's Information Criterion) low.

To answer Research Question 4.2, I calculated the gap size and include this as a variable in the logistic regression model to study the effects on build success probability.

I used R [2] and the Design package [1] for logistic regression analysis.

6.3 Results

In the RTC repository, I analyzed 191 builds; of these builds, 60 were error builds, and 131 were OK builds. Table 6.1 displays summary statistics per build. The table displays the number of authors, files, change sets, and work items per build, the build date range (on average), and the calculated unweighted congruence, weighted congruence, and gap size. Figure 6.8 displays histograms for unweighted congruence, and Figure 6.9 shows histograms for weighted congruence. The histograms compare the frequencies for each type of congruence for all builds, the OK builds, and the error builds only. There were some minor differences between unweighted and weighted congruence values; weighted congruence, for instance, largely reduced the number of "fully" congruent situations where congruence is 1.

The congruence values were low on average. The unweighted congruence had a mean value of 0.331, and the weighted measure had a mean value of 0.196, meaning that about one-third and one-fifth of the technical relationships were satisfied by social relationship, respectively. Over 75% of the builds had a weighted congruence value of less than 0.25.

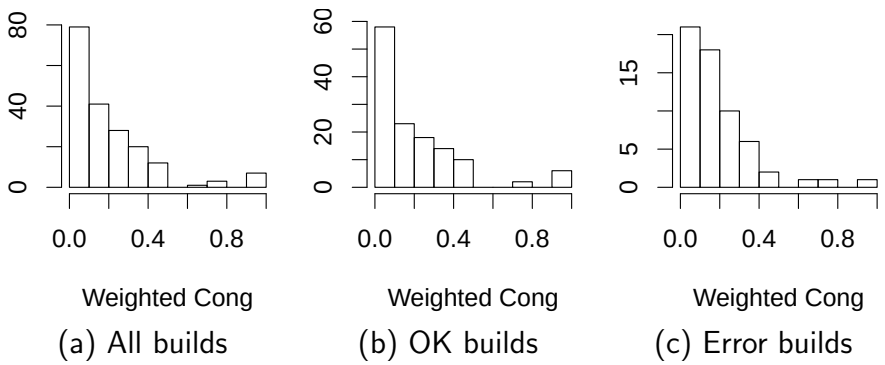


Figure 6.9: Distribution of Weighted Congruence Values

	Min	Median	Max	Mean
Authors	2	17	44	18.62
Files	5	131	3101	342.3
Change Sets	4	34	226	54.2
Work items	4	34	182	48.3
Build date range (days)	0	345	361	319.2
Unweighted cong.	0	0.21	1	0.331
Weighted cong.	0	0.15	1	0.196
Gap size	-0.083	0.190	1.00	0.317

Table 6.1: Summary Statistics

I calculated pairwise correlations between the variables weighted congruence, unweighted congruence, number of authors, number of files, number of change sets, number of work items, and build date (Table 6.2). A high positive number means that the variables are heavily correlated—that is, an increase in one variable's value also increases the other. Similarly, a low negative number means that the variables are negatively correlated. There was a strong correlation between weighted and unweighted congruence, which was expected; a strong correlation between change sets and authors; and a weak correlation between authors and files. To avoid multicollinearity problems in the data, I chose to remove change sets from the logistic regression analysis because, due to the enforced processes in RTC, I knew that there was exactly one author per change set, and thus there were at least as many change sets as authors per build. Note that weighted congruence and unweighted congruence do not appear in the same model and are thus not subject to multicollinearity problems.

To assess the fit of the logistic regression models, I used the Nagelkerke pseudo-R^2 and AIC. R^2 shows the proportion of variability explained by the model, and AIC is a measure of how well the model fits the data. Ideally, R^2 is high and AIC is low. The current model contains 19 variables and has an R^2 of 0.581 and 0.548 for unweighted and weighted congruence models respectively. I compared the model in Table 6.3 to

	2.	3.	4.	5.	6.	7.
1. Weighted	0.77	-0.22	-0.14	-0.12	0.05	0.02
2. Unweighted	–	-0.27	-0.22	-0.33	0.08	0.19
3. Authors		–	0.41	0.76	0.10	-0.30
4. Files			–	0.37	0.08	-0.20
5. Change sets				–	0.02	-0.38
6. Work items					–	0.04
7. Build date						–

Table 6.2: Pairwise Correlation of Variables per Build

		Unweighted		Weighted	
Model	Variables	AIC	R^2	AIC	R^2
Every interaction	27	188.6	0.595	196.7	0.559
Main effects only	7	213.2	0.269	214.2	0.263
Every interaction, no congruence	15	190.1	0.479	190.1	0.479
Our model	**19**	**175.8**	**0.581**	**183.4**	**0.548**

Table 6.3: Model Comparison

a model containing every first-order interaction effect with 27 variables, a model that contains the 7 main effects only, and a model with the interaction terms, but without the congruence variable (Table 6.5). I found that 19 variables is optimal and that removing further variables lowered the R^2 value while raising the AIC.

I observed also that the unweighted congruence model and the weighted congruence model are very similar, with a difference in Nagelkerke R^2 of only 0.030.

	Unweighted			Weighted		
Variable	Coef.	S.E.	p	Coef.	S.E.	p
Intercept	-0.5459	0.4663	0.2417	0.3289	0.4729	0.4867
congruence	**6.3410**	**1.6262**	****0.0001**	3.6699	2.3013	0.1108
authors	**-1.9759**	**0.5310**	****0.0002**	**-2.0176**	**0.5802**	****0.0005**
files	**-1.0734**	**0.4561**	***0.0186**	**-1.1169**	**0.5099**	***0.0285**
workitems	-0.1456	0.2355	0.5363	-0.1365	0.2343	0.5602
build type=I	**2.1533**	**1.0526**	***0.0408**	1.2777	1.0172	0.2091
build type=N	4.6833	200.7587	0.9814	2.7593	192.5236	0.9886
build date	-0.6560	0.6709	0.3282	-0.1352	0.7133	0.8497
congruence * build type=I	**-9.2151**	**2.5572**	****0.0003**	**-6.2748**	**2.9664**	***0.0344**
congruence * build type=N	-7.7308	91.8053	0.9329	-7.2024	188.1811	0.9695
congruence * build date	**-5.1266**	**1.9290**	****0.0079**	**-5.5670**	**1.9760**	****0.0048**
authors * build type=I	1.2688	0.7028	0.0710	1.1852	0.7370	0.1078
authors * build type=N	105.4123	535.8792	0.8441	103.0155	521.0808	0.8433
authors * build date	-0.6061	0.3616	0.0937	-0.6004	0.3576	0.0932
authors * files	0.7663	0.4289	0.0740	0.4979	0.5232	0.3414
files * build type=I	1.0920	1.1838	0.3563	1.7042	1.4099	0.2267
files * build type=N	-37.9274	199.2314	0.8490	-36.5156	192.3382	0.8494
workitems * build date	**0.8040**	**0.3003**	****0.0074**	**0.8909**	**0.3466**	***0.0102**
build type=I * build date	**2.6442**	**0.7678**	***0.0006**	**1.8627**	**0.7621**	***0.0145**
build type=N * build date	84.7252	344.8129	0.8059	84.3117	341.4988	0.8050
Model likelihood ratio	101.92	$p \approx 0$	$R^2 = 0.581$	94.36	$p \approx 0$	$R^2 = 0.548$
	191 observations			191 observations		

Build build type is set to continuous

$^*p < 0.05$; $^{**}p < 0.01$

Nagelkerke is used as the pseudo-R^2 measure

Table 6.4: Logistic Regression models predicting build success probability with main and interaction effects

	Unweighted			Weighted		
Variable	Coef.	S.E.	p	Coef.	S.E.	p
Intercept	0.5265	0.3040	0.0833	1.00416	0.2754	0.0003
congruence	0.9371	0.6807	0.1686	-0.85692	0.8544	0.3159
authors	**-0.5702**	**0.2003**	****0.0044**	**-0.64635**	**0.2023**	****0.0014**
files	**-0.6398**	**0.2477**	****0.0098**	**-0.70618**	**0.2571**	****0.0060**
workitems	-0.1755	0.1713	0.3055	-0.13229	0.1689	0.4335
build type=I	0.1693	0.4269	0.6917	0.06128	0.4154	0.8827
build type=N	0.2133	0.7791	0.7842	0.29418	0.7811	0.7065
build date	-0.1331	0.1821	0.4649	-0.11291	0.1819	0.5349
Model likelihood ratio	40.59	$p \approx 0$	$R^2 = 0.269$	39.52	$p \approx 0$	$R^2 = 0.263$
	191 observations			191 observations		

Build type is set to continuous

$^*p < 0.05$; $^{**}p < 0.01$ Nagelkerke is used as the pseudo-R^2 measure

Table 6.5: Logistic Regression Models Predicting Build Success Probability With Main Effects Only

6.3.1 Effects of Congruence on Build Result

To answer Research Question 4.1, I performed a logistic regression analysis between the variables of the builds and the build outcome (Table 6.4).

The result of logistic regression indicated that the following effects were significant for both unweighted and weighted congruence models: The congruence × build type effect, the congruence × build date interaction effect, the number of work items × build date interaction effect, and the build date × build type effect. In addition, the number of authors and the number of files were significant main effects, although their coefficients were lower than the interaction effects involving congruence. I also identified unweighted congruence as a significant main effect in the unweighted congruence model. Because both unweighted congruence and weighted congruence are identified as significant when interactions are considered, Hypothesis 4.1 which stated "increasing socio-technical congruence correlates to increased team performance" holds. However, because of the presence of interaction effects, it must be noted that this hypothesis holds only under certain contexts.

In the next section I discuss the main effects and interactions effects that involve congruence affecting build probability. I discuss the effects of the non-congruence effects, including the authors, files, work items × date interaction effect and the date × nightly build effect in Section 6.3.3.

Effects of interactions involving congruence

The type × congruence interaction effect, the date × congruence interaction, and the type × date effect were each significant in the model (Table 6.4). I plotted in Figures 6.10 and 6.11 the effects of weighted congruence vs. probability of build success at the 10% date quantile (2008-01-25), at the 25% date quantile (2008-05-14), the 50% date quantile (2008-06-07), and the latest build (2008-06-26).

For continuous builds in weighted congruence (Figures 6.11, in grey), an increase in congruence correlated to build success probability. The build date as the project ages appeared to decrease the probability of build success for high-congruence builds more than low-congruence builds (Figure 6.11(d)). Note that, in the unweighted congruence model (Table 6.4), the effect of unweighted congruence on continuous builds was significant, and that increasing congruence also increased the probability that a continuous build succeeds. However, the effect of weighted congruence in continuous builds is not as large as in unweighted congruence, nor is it considered statistically significant.

For integration builds (Figures 6.11, in black), an increase in congruence decreased build success, with the exception of the 2008-01-25 build (Figure 6.11(a)). In in the 2008-01-25 build, low congruence led to low build probability, but high congruence had high build probability. As the project aged, this trend reversed and congruence was clearly inversely related with build success probability (Figure 6.11(d)).

The effect of congruence was totally opposite for continuous builds and integration builds in both unweighted and weighted congruence. Based on Figure 6.10(d),

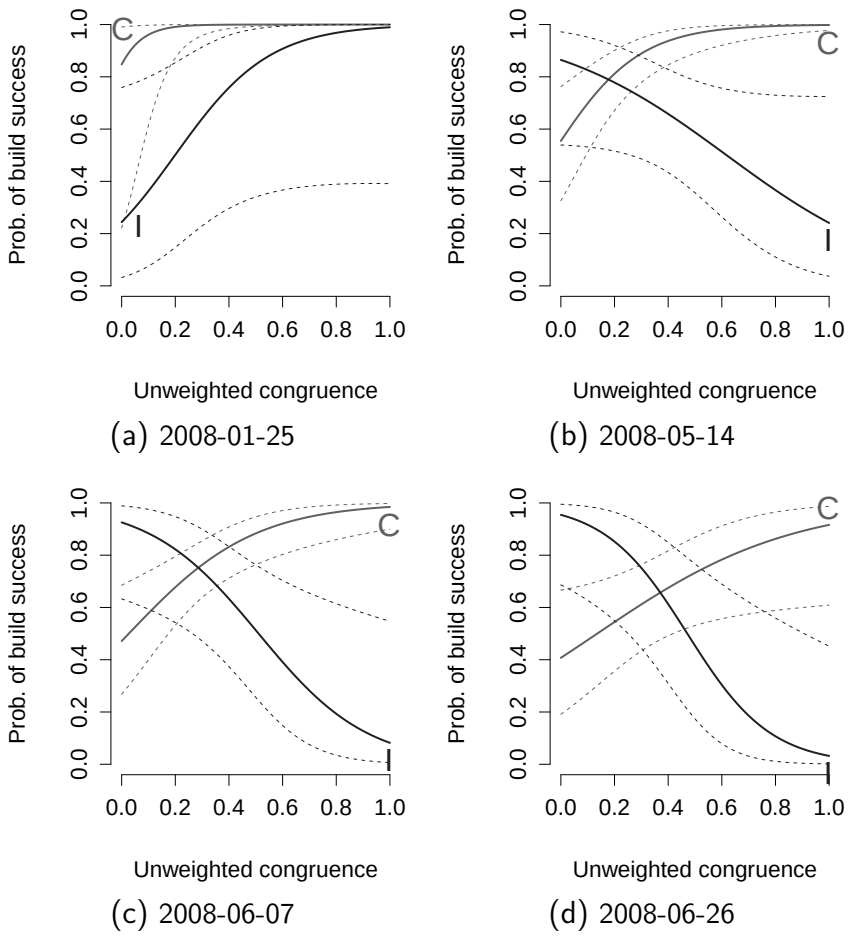


Figure 6.10: Estimated Probability of Build Success for *Unweighted Congruence* and *Continuous builds C* or *Integration builds I* Over Time, Adjusted to Authors ≈ -0.156 (17 authors), Files ≈ -0.352 (131 files), Work Items ≈ -0.399 (34 work items)

increasing unweighted congruence significantly improved the continuous build success rate. However, increasing both types of congruence significantly decreased the integration build success rate.

6.3.2 Effect of Gap Size on Build Result

To answer Research Question 4.2, I fitted a logistic regression model to determine the effect of the mean gap size on the build success probability. The mean gap size appears in Table 6.1 and Figure 6.12.

I built logistic regression models based on the model in Table 6.4 using the gap size measurement. I retain every significant interaction from the previous weighted congruence logistic regression in Table 6.4 and report the odds ratio. I compared two different models: Model G1 contains the gap size with weighted congruence as a

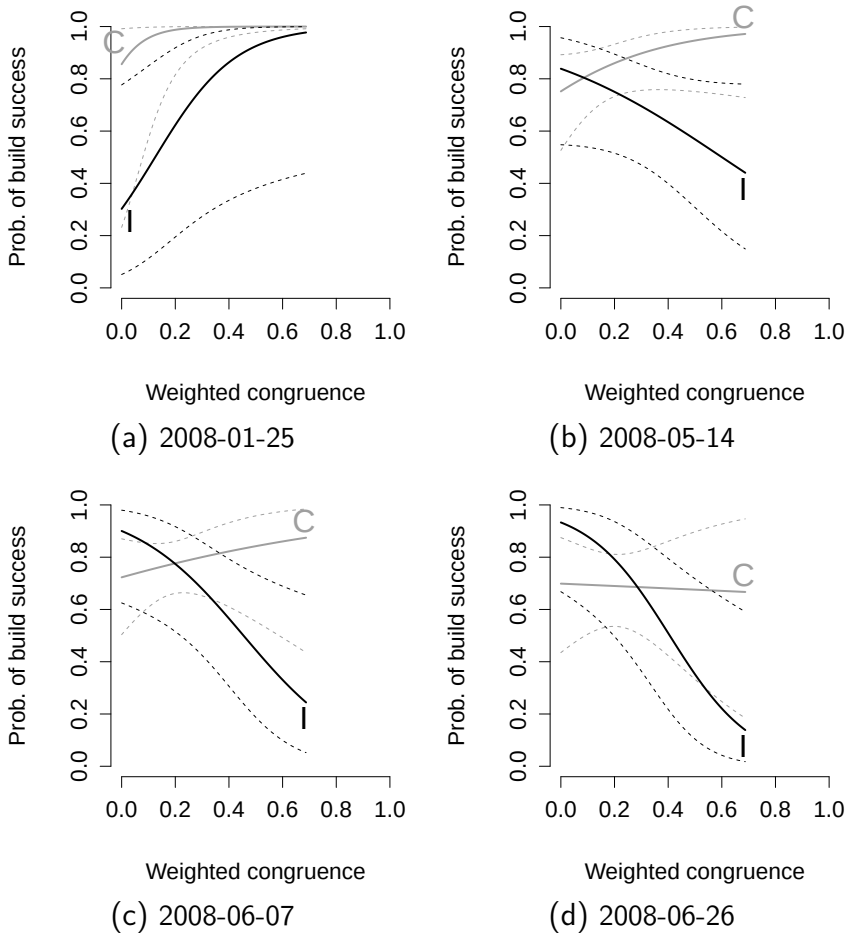


Figure 6.11: Estimated Probability of Build Success for *Weighted Congruence* and *Continuous Builds C* or *Integration Builds I* Over Time, Adjusted to Authors ≈ -0.156 (17 authors), Files ≈ -0.352 (131 files), Work Items ≈ -0.399 (34 work items)

variable, and Model G2 contains the gap size without weighted congruence. I do not use unweighted congruence as it is not possible to calculate gap size with unweighted congruence.

The effect of gap size on build result was significant in both models (Table 6.6). This indicated that increasing the gap size significantly increased the odds that an OK build would occur (Figure 6.13). This is the opposite of Hypothesis 4.2, which stated, "Decreasing the gap size correlates to increased team performance". This suggests that if the gap size is large, the build success probability increases.

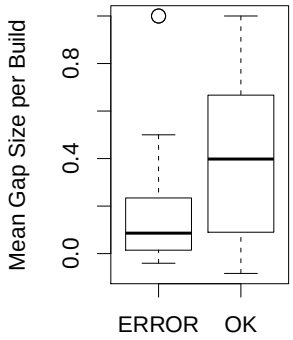


Figure 6.12: Mean Gap Size per Build

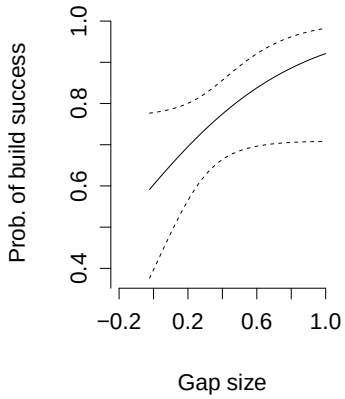


Figure 6.13: Effect of Gap Size on Build Success Probability, Model G1.

6.3.3 Social and Technical Factors in RTC Affecting Build Success and Congruence

In light of my results, I examined not only the number of work items × date significant interaction found in Section 6.3.1, but different social and technical factors that may have affected congruence and build success probability to find explanations for the interactions between socio-technical congruence and build success probability in RTC. Specifically, I examined the effect of build date on work items, coordination around fully-congruent builds and incongruent builds, and the effects of commenting behaviour on builds.

Other Effects on Build Success

Authors For both weighted and unweighted congruence, as the number of authors involved in a build increased, the probability that the build succeeds decreased. The build probability was significantly lowered after more than 15 authors were involved in the build (Figure 6.14(a)). When over 30 authors were involved in the build, the

	Model G1	Model G2
Intercept	0.95	1.32
authors	0.43	0.60
files	0.63	0.63
workitems	0.75	0.85
build type=I	4.22	1.31
weighted cong	8.41	-
gapsize	7.71	8.71
builddate	1.81	0.59
authors * build date	0.40	0.74
workitems * build date	2.75	1.83
build type=I * build date	3.54	2.52
build type=I * weighted cong	0.01	-
weighted cong * build date	0.00	-

Table 6.6: Odds Ratio for Gapsize Models

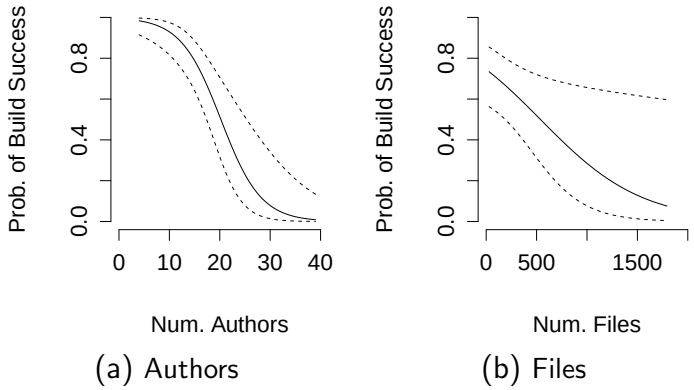
estimated build success probability falls under 10%.

Files For both weighted and unweighted congruence, as the number of files involved in a build increased, the probability that the build will succeed decreased (Figure 6.14(b)).

Build Date and Work items The work items × date interaction was significant in both weighted and unweighted congruence. Early in the project, as the number of work items increased, the probability of build success decreased (Figure 6.15(a)). As the project aged, this trend reversed and as the number of work items increased, the probability of build success increased as well (Figure 6.15(b)). According to the coefficients in Table 6.4, this effect on build success probability was not as strong as the authors main effect or the files main effect.

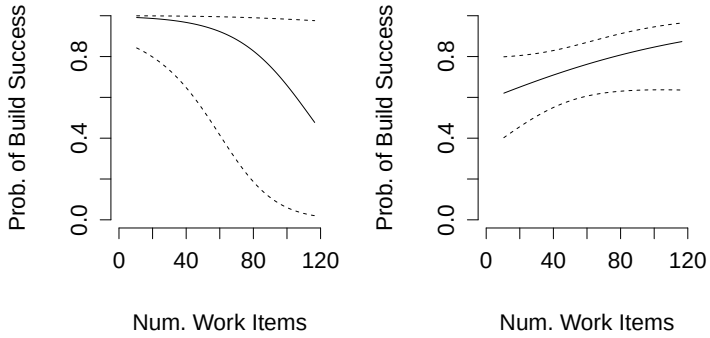
Commenting Behaviour by Change Set Authors

As congruence expects those who contribute technical work to a project to also communicate changes, I decided to examine commenting behaviour of change set contributors. I examined whether a work item related to a change set had a corresponding comment posted by the change set's author. I also identified if the work item/change set pair was involved in an OK build or an Error build. Note that a work item/change set pair could occur in multiple builds. I considered only comments posted before the build was started. These results are shown in Table 6.7. The build result rates did not appear to be affected by whether the change set contributor commented: 43% of the builds resulted in error when the contributor commented, and 41% of builds resulted in error when the contributor did not comment. A χ^2 test on this table did not show a significant difference between the factors at the 0.05 threshold ($\chi^2 = 3.217$, df = 1, p=0.073). Overall, in 76% of the change set/work item pairs, the change set



(a) Authors (b) Files

Figure 6.14: Estimated Probability of Build Success for *Authors* and *Files*, Weighted Congruence. Adjusted to Work Items ≈ -0.399 (34), Authors ≈ -0.156 (17), Files ≈ -0.352 (131), W. Congruence ≈ 0.1446, Type=cont, Date=2008-06-26



(a) 2008-01-25 (b) 2008-06-26

Figure 6.15: Estimated Probability of Build Success for *Work Items* and *Date*, Weighted Congruence. Adjusted to Authors ≈ -0.156 (17), Files ≈ -0.352 (131), W. Congruence ≈ 0.1446, Type = cont

contributor also posted a comment in the associated work item.

Examining Extreme Congruence Values

I am interested in the differences between high-congruence builds and low-congruence builds. I furthered this investigation by looking at builds that had extreme values of congruence: zero, where absolutely no technical relationships were satisfied with communication, and one, where every technical relationship was satisfied with communication. I chose to investigate the extreme cases to see if there were differences in the way people coordinated in fully-congruent builds, and in incongruent builds. Table 6.8 shows the number of OK and error builds that occurred when congruence was equal to one, and equal to zero. The weighted builds with full congruence were a subset of the unweighted builds with full congruence.

Result	Change sets w/o comment by contributor	Change sets w/ comment by contributor	Total
OK	1278 (14%)	3956 (43%)	5234 (57%)
ERR	908 (10%)	3076 (33%)	3984 (43%)
Total	2186 (24%)	7032 (76%)	9218 (100%)

Table 6.7: Number of Change Sets in which a Contributor Commented in the Corresponding Work Item

		Congruence	
		1	0
Unweighted	OK	26	30
	ERR	2	2
Weighted	OK	6	30
	ERR	1	2

Table 6.8: Number of Builds with Congruence Values 0 and 1

To determine if the presence of commenting affected the builds, I examined the number of comments on work item–change set pairs in builds with extreme unweighted congruence values. My results are shown in Table 6.9. Build success probabilities improved with respect to builds that have no comments, though work items with no comments were in the minority.

Of note was the high number of comments on work items that had zero congruence. This indicated that individuals who had no technical relationship to the work item were commenting on the work item.

		Num. of Pairs		Success rate	
Congruence		1	0	1	0
No comments	OK	42	143	49%	69%
	Error	43	64		
Comments	OK	610	445	68%	69%
	Error	290	199		
Total	OK	652	588	66%	69%
	Error	333	263		

Table 6.9: Number of Work Items-Change Set Pairs with Comments and Build Success Probabilities for Congruence 0 and 1

I manually inspected the work items with extreme amounts of congruence, reading the comments for any topic differences in the content discussed. Unfortunately, there were no obvious qualities between comments made in a build with a congruence of

zero, and comments made in a build with a congruence of one. In both builds, individuals discussed technical implementation details, provided updates to colleagues, or requested assistance from colleagues.

6.4 Discussion

The concepts illustrated in Conway's Law, as well as previous empirical work on socio-technical congruence lead us to expect that team members must coordinate according to technical relationships suggested by technical dependencies in order to build software effectively. In this empirical investigation, I applied socio-technical congruence to study coordination and its relationship to team performance, which was conceptualized by build success probability. I applied a modified weighted congruence measurement to study also how the size of a coordination gap affects build success probability, and investigated how social and technical factors in RTC affect congruence and builds.

Overall, I found that the average congruence across builds was very low—only 20–30% of the technical relationships in the project were fulfilled with a social relationship. Even in the cases where there is zero congruence, the build result was an OK build in over 90% of the observed cases.

Research Question 4.1: Does socio-technical congruence have an effect on build success probability? I hypothesized that socio-technical congruence would correlate with team performance, conceptualized as build success probability. I found in this study that there was an interaction effect involving congruence and build type on build success probabilities (Section 6.3.1), suggesting that there is an effect but only in certain contexts. For continuous builds, increasing congruence improved the chance of build success in continuous builds and can actually decrease build success probability in integration builds (Figures 6.11). High unweighted congruence significantly improved continuous build success probability, and both unweighted and weighted congruence significantly reduced integration build success probability.

Research Question 4.2: Does gap size have an effect on build success probability? The gap size is a representation of whether enough coordination occurred to fulfill multiple technical relationships. If two developers have multiple dependencies on each other, one would expect them to coordinate more often as well. I hypothesized that a small mean gap size would increase the probability of successful builds and that a large mean gap size would decrease the probability of a successful build. Instead, I found that as the mean gap size increases, the build success probability also increases (Figure 6.13).

Below I discuss the reasons for these observed results based on my knowledge of RTC.

6.4.1 Project Context and Socio-technical Coordination

The project context affected my socio-technical congruence measurement conceptualizations.

The overall congruence for the majority of builds was low: over 75% of builds had a congruence of less than 0.25 (Section 6.3.1). Despite low congruence, the RTC team was able to successfully build its software in many situations. The fact that the change set author commented on a work item related to the change set does not appear to affect build results either (Section 6.3.3), suggesting that changes did not need to be explicitly communicated for every build.

When I examined extreme congruence values, I observed 85% build success probability when weighted congruence is 1 and 93% build success when weighted congruence is 0 (Section 6.3.3). If socio-technical congruence is a measure of coordination quality in software, and builds rely on coordination quality to be successful, then there must be reasons why builds can succeed even when the congruence is zero.

First, because RTC is a highly-distributed project, the product under development used a modular design [145] and thus is affected less by dependencies. Second, team members in RTC did not conduct all of their coordination through *explicit communication* even though work item inspection and discussion with developers indicated that the RTC corporate culture focused on the work item as their base for communication. Rather, they used the *shared workspace* that incorporates cues from the environment and from peers in order to address technical issues. Both of these effects may contribute to congruence being lower than expected.

RTC Supports Explicit Communication

The RTC team members used the RTC environment extensively to communicate with each other. I was informed that RTC team members rarely use private email, and my inspection of the mailing list reveals that its primary purpose is for announcements such as server outages rather than for discussing technical work.

This left the RTC work item comment system and instant messaging as methods for communication, as well as the phone and internal face-to-face meetings. I learned that while face-to-face interaction is efficient for solving local issues, it did not benefit remote teams, and the RTC team as a whole encouraged every team member to record face-to-face discussions as comments for the purpose of archiving and sharing information.

However, explicit coordination had a cost. There was evidence that involving too many authors in the same build also reduced the build success when using a weighted congruence conceptualization (Figure 6.14(a)); the effect for unweighted congruence is similar. The overhead required to coordinate many people may interfere with the ability of the team to build the project successfully, suggesting that there was a limit before a developer is overloaded with information.

RTC is a Shared Workspace

The RTC client software helped a developer acquire and maintain *environmental awareness* of what was going on in the project by providing access to a shared workspace. Much of the work was centred around the RTC technical entities, which include plans, source code, work items, and comments. RTC's awareness mechanisms featured a developer-centred dashboard that reports changes to the workspace, built-in traceability, user notifications, regularly-generated reports, and an optional web browser interface. For example, when a change set is created, it is attached to a work item, thus ensuring that people who are involved with the work item receive notification of this change set. These automatic notifications cut down the amount of explicit communication and allow people to coordinate implicitly.

Coordinating using the workspace is well-known in the computer-supported cooperative work domain [192]. Open-source developers, in particular, coordinate around source code [34] and mailing lists [103, 159] because there is little opportunity for face-to-face interaction. RTC shares many characteristics with open-source development, such as a distributed team and a transparent development process.

In light of these results, I believe that, using my conceptualization, the RTC team required a congruence of only 0.2–0.3 for their tasks to be completed. Much of the need for explicit, point-to-point communication was mitigated by implicit communication and the use of the workspace to coordinate. I expect that the remaining congruence is covered through the RTC workspace, and through face-to-face communication, instant messenger, and phone communication. It is also a possibility that software builds required less coordination than conceptualized in the study. Though my congruence value appeared low for the RTC team, the coordination in reality may be higher. Future studies should keep in mind that congruence may be lower than expected because of conceptualizations that cannot include every type of coordination in a project.

6.4.2 Coordination and Geographic Distribution

As RTC is a distributed team, geographic distribution had an effect on team performance, though the RTC environment helps mitigate some of these effects [166].

I learned from the RTC project that continuous and nightly builds should involve mainly a co-located team, and that integration builds involve multiple components from RTC teams in different locations. My results suggested that congruence best benefits builds that occurred within the same logical team. However, the design of my study does not allow me to draw a firm conclusion about the influence of co-location and congruence on build success probability.

It appears that involving too many individuals when coordinating the activities of various teams may harm build success due to information overload [69], especially when the team members are distributed. To negate this effect, development leaders and build managers that have an overall view of the project are suited to coordinate teams to ensure build success [118].

6.4.3 Communication Between Individuals Occurs When Problems Arise

My results indicated that even when congruence gaps are bridged, there is a high probability that a build still fails.

I have observed that, due to awareness through RTC, the RTC team was able to react to situations with multiple technical dependencies by communicating quickly, a trait that has been previously observed among open-source developers [159]. Contributors were likely to comment on their own change sets (Section 6.3.3).

The RTC team was aware of builds that require high coordination, and are able to address the situation by coordinating with comments. If a work item was particularly complex during development, a developer would post comments on a work item requesting expertise and informing others about unanticipated problems. This amount of written communication may not have occurred in an OK build simply because the level of coordination was not necessary; in essence, not every technical dependency requires explicit coordination to bridge that gap. There was also the possibility that developers coordinated on difficult builds using methods that are not captured by my conceptualizations such as phone or face-to-face. An error build may have been complex, requiring extensive coordination among developers and therefore shrinking gap sizes in error builds. More study of how developers coordinate under "problematic situations" is warranted.

6.4.4 Project Maturity and Build Success

I found that early builds exhibited a different type of relationship between congruence and build success probability than later builds (Section 6.3.1). Over the course of the study, I observed 13 internal milestones; the last milestone in the observed builds was a public beta release for end users.

Build success probability decreased significantly over time for continuous builds and stayed roughly the same for integration builds (Figures 6.11). However, the early builds in the project behaved contrary to later builds in the project (Figure 6.11(a) and 6.11(a)). The RTC software early in its lifetime was in a state of change. Integration builds were not a priority, and features were being added to the project. This means that dependencies, as well as the expertise among team members were both changing rapidly, making it difficult to solidify technical relationships.

In addition to interactions between congruence and type, and congruence and date, I observed a significant interaction effect between build date and work items. I found that early in the project (Figure 6.15(a)), builds with large numbers of work items have a high probability of failing, but late in the project (Figure 6.15(b)), these builds succeeded. Because the latest release was focused on a public release, a build linked to numerous work items may indicate that a bug is highly problematic or a feature is highly desired, and therefore received more attention. This is similar to the effect discussed with gap size (Section 6.4.3), where an error build required more

coordination from involved developers.

6.4.5 Threats to Validity

Socio-technical congruence can be difficult to compare between studies. Conceptualizations, such as technical dependencies and social relationship, vary from project to project. Some project-specific conceptualizations I used are the build as a measure of success, the fact that only files that are touched by more than one individual qualified as having dependencies on each other, and the work item comments connecting people in a clique. Because of the context-sensitive nature of socio-technical congruence, especially with respect to the construction of communication and dependencies, it is difficult to apply socio-technical congruence as a benchmark for coordination. Having an understanding of the context of the project is extremely important when interpreting results obtained from socio-technical congruence calculations.

The technical relationships were likely overestimated due to the modular nature of RTC. As I was studying a distributed project that follows a transparent development process, RTC used a modular design [145] and a change in a file in a change set may not necessarily be dependent on other changes within the same file. I observed evidence in my study that the dependencies among change sets as well as the attached work items may not be as strong as I have originally believed.

The conceptualization of social relationship underestimated the amount of communication that truly occurs in the project. I relied on repository data in RTC and were unable to conceptualize forms of communication such as instant messenger, phone calls, and face-to-face interactions. Due to geographical distance and time zone differences, the RTC team's primary mode of collaboration was the work item comment system, but I made a number of assumptions about commenting behaviour.

First, I assumed that everyone involved in the work item reads every comment. Second, I did not take into consideration any additional coordination that may occur from a silent onlooker reading the comments. I believe intuitively that the first effect is greater than the second. This may help explain part of the effect of the unintuitive gap size result; rather than the gap being actually large in reality, it is large in my conceptualization because social relationship outside of commenting is not recorded.

Another threat to validity is that the data does not cover every build executed in the lifetime of RTC. RTC did not keep a full archive of build results, and as a consequence, I did not have a full population from which to draw data from. The threats are particularly high for early data points. The large confidence intervals in some of the builds, namely nightly builds, reflect this lack of data, thus I hesitate to draw conclusions based on early builds and nightly builds.

Finally, this study is a single empirical investigation and the results are not generalizable, nor can they be directly compared to existing studies due to the differences in the projects under examination. However, I believe that this study advances the theoretical and the empirical examination of socio-technical congruence.

6.5 Modifications to the Socio-technical Congruence Model

In addition to answering Objective 4, applying the socio-technical congruence measurement to RTC has inspired a number of requirements for the socio-technical congruence framework. I highlight the contributions from the RTC study to the model in Figure 6.16.

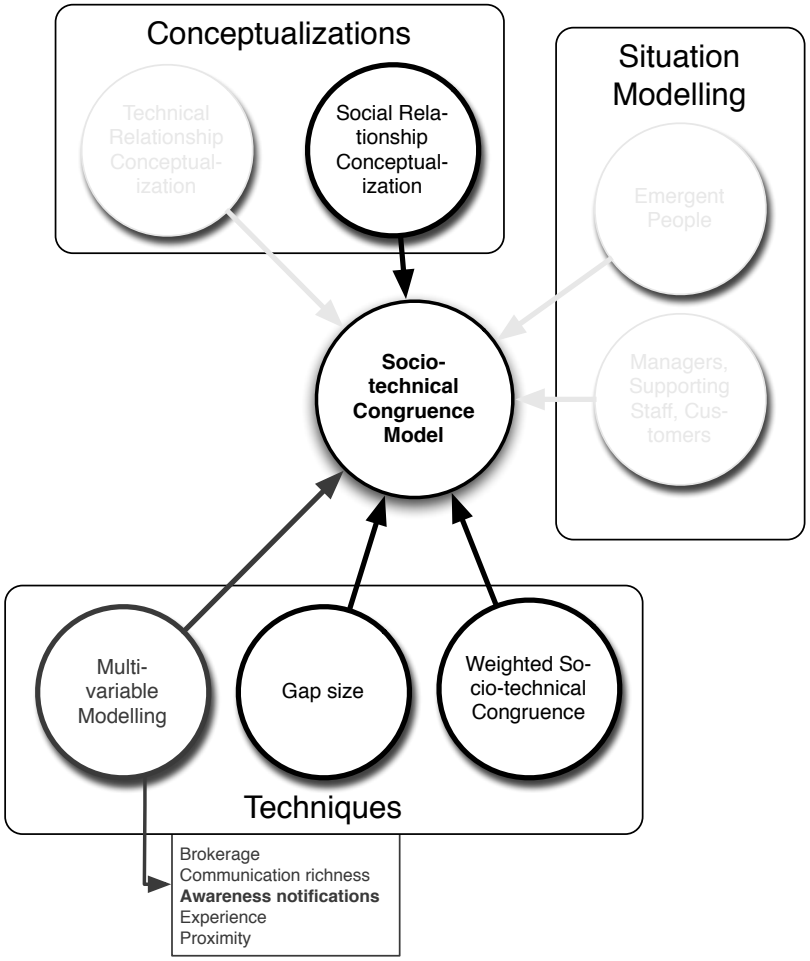


Figure 6.16: Elements From the Socio-technical Congruence Study that Contribute to the Socio-technical Congruence Model

This study has shown that high socio-technical congruence does not always lead to good outcomes, and that congruence depends highly on the nature of the activity. This study in particular revealed that congruence led to better probability of success for continuous builds, but not for integration builds. There are fewer integration builds than continuous builds, which suggests that high congruence can lead to improved outcomes for some activities, but suggests that there is an upper limit to the

amount of coordination that can occur. The context and the type of the activity may determine the effect of socio-technical congruence on the outcome.

The choice of technical entity and technical dependency is extremely important when applying socio-technical congruence. The **social relationship that is being collected must match the technical relationship** that is derived based on the technical dependencies. One reason why the congruence was low in this study was because the developers did not feel the need to bridge every technical relationship with work item comments—rather, the use of awareness mechanisms through the RTC environment and through source code was sufficient. Thus, it would not be a good idea to recommend that the RTC team achieve full congruence, at least not using the conceptualizations of socio-technical congruence presented in this study.

Because of the use of both awareness through the integrated development environment, and work item comments, this study highlights the need to **represent multiple variables** in socio-technical congruence. In the RTC empirical investigation, I represented communication through work item comments, but there was evidence that awareness through the RTC environment was influential. Being able to model awareness relationships in addition to comment relationships, while keeping the data separate, would allow flexible calculations to determine if communication, or awareness, was more able to satisfy the files-changed-together technical relationships in RTC. A general model of this in a socio-technical congruence model is necessary. Many contextual factors, such as the support provided by RTC for explicit communication, the automatic notifications sent to team members, and the maturity of the project and its team members, affect software engineering outcomes, and these multiple factors must be taken into account in conceptualizations of socio-technical coordination.

6.5.1 The Use of the Weighted Congruence Measure

I presented a weighted congruence measure which provides measurements such as gap sizes and relationship strengths that were not available by using unweighted congruence. Although the weighted congruence measure yielded similar results to the unweighted congruence measure, the weighted congruence model appeared to lower the threshold of congruence by smoothing out the index values into a more balanced distribution compared to unweighted congruence (compare Figures 6.8(a) and 6.9(a)). This led to the unweighted congruence model having a significant main effect on continuous build success, whereas the weighted congruence model did not have such an effect. This shows that **weighted congruence is a more conservative measure than unweighted congruence**. Depending on the objective of the analysis, the unweighted congruence may represent an upper range of potential congruence in a project, whereas the weighted congruence tends to be less likely to report perfect congruence and may represent an average scenario.

Consequently, the unweighted congruence measure is a less conservative measure than weighted congruence. One may want a less conservative measure if a team is

observed to work well together in a way that cannot be represented by the conceptualization. Weighted congruence measure is more conservative and is appropriate in projects where contextual information about the team is not known. In addition, weighted congruence provides **gap size**, which have been shown to reveal some potential phenomenon for investigation. Thus, the measure provides additional information without sacrificing the capabilities of unweighted congruence.

Chapter 7

Revisiting the Socio-technical Congruence Model

The findings from the empirical investigations of socio-technical coordination reveal how conceptualizations of social relationships and technical relationships in my socio-technical congruence model can be improved. I refine the model that I developed from Chapter 3 to include results of my empirical investigations. In particular, the results of the study refine the conceptualization of social and technical relationships, provide guidelines about how to conceptualize different aspects of coordination, and provide insight into exceptional situations and circumstances around social interactions in software development teams.

> **Objective 5: To improve the socio-technical congruence model developed in Objective 1 by incorporating the results of the empirical investigations from Objectives 2, 3, and 4.**

The Ship studies revealed a number of additional factors such as different aspects of awareness, communication, and emergent people that are not considered in socio-technical coordination studies. The application of the socio-technical congruence model to the RTC project applied some proposed modifications, such as the lack of weights in gaps. The RTC study also emphasized areas of the model that could be improved, and provided some preliminary evidence that gap size and weighted socio-technical congruence measurements can reveal valuable research findings.

One of the contributions of the model that I develop is that such a model brings to the forefront a discussion about the conceptualizations that can be used in socio-technical congruence, and the situations that socio-technical congruence should be able to represent. Taking a higher-level perspective of socio-technical coordination allows us to examine fine-grained interactions at the task level while still ensuring that relevant team members are identified and appropriately involved in coordination.

Figure 7.1 is the complete picture of the elements that were added to the model based on the studies.

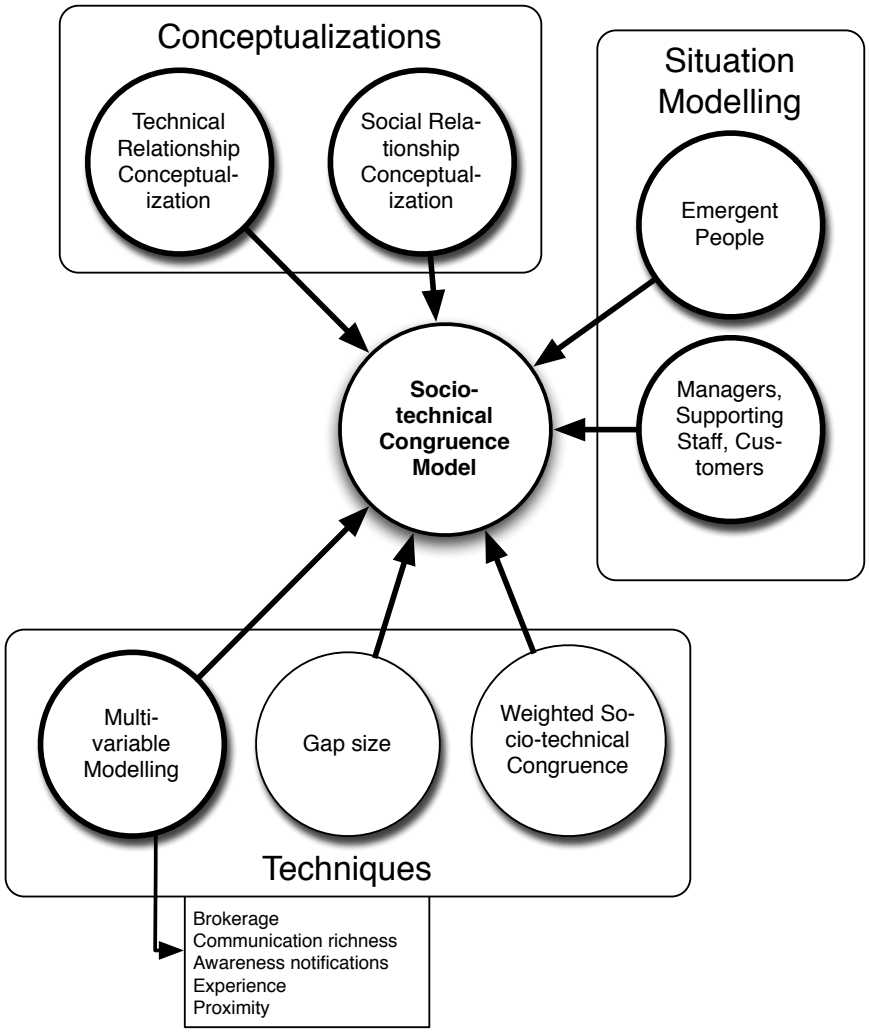


Figure 7.1: Elements of the Socio-technical Congruence Model, Completed with Contributions

7.1 Additions Within the Model Elements

The socio-technical congruence model described the entities in the model, including technical and social relationships and a connecting entity. The model elements, *conceptualization*, *techniques*, and *situation modelling* (Chapter 3) outline where extensions to the model can be added and refined. Modifications to the definition and representation of technical relationships or social relationships belong to the "conceptualization" element. Additions to the calculation of alignment or details regarding the meaning of gaps belong to the "alignment" element. Situations that are unique or exceptional, such as communication patterns over time, belong to the "situation modelling" element.

There are five areas within these elements that I extend based on the results

of the empirical investigations in this research. These areas are the refinement of technical relationships, the refinement of social relationships, incorporating emergent people, incorporating managers, supporting staff, and customers, and multi-variable modelling. I also discuss how to use gap size and weighted socio-technical congruence in light of the application to the RTC empirical investigation.

7.2 Refining Technical Relationships

One of the research goals of this dissertation was to refine the conceptualization of **technical relationships** and **social relationships**, with a particular focus on the social aspect.

The technical entity, from which technical relationships are derived, is the basis of socio-technical congruence. Consequently, select technical entities that reflect the technical relationships of team members.

The roles of the individuals that participate in the project may affect your choice of technical entity. A technical entity can be more than files in a software project—the technical entities can be groups of files, or they may be higher-class entities, such as models, requirements, or features. Dependencies must be identifiable between the technical entities. In addition, relationships between individual contributors and the technical entity must also be identifiable.

7.2.1 Satisfying Technical Relationships

Not every technical relationship requires explicit communication. When considering the technical entities to analyze, consider whether the technical relationships that are calculated based on the technical entities and technical dependencies require people to coordinate explicitly through communication, or if they can coordinate implicitly through the environment. Certain types of technical dependencies require more action from the team members than other types.

The comment analysis in RTC highlighted that many of the builds are successful without any communication. Related work by Bolici et al. [34] suggests that developers can coordinate not only through communication, but through artifact "traces". Studying the work items in RTC revealed that there were no feature-level dependencies in work items from builds—that is, the tasks for each work item were generally self-contained. This corroborates the findings of MacCormack et al. [145] that identified that projects with characteristics similar to RTC have a modular architecture. This would reduce the need for coordination with others.

Thus, it is important to consider whether two people who work on dependent modules actually need to communicate. Software information hiding attempts to hide most changes [174] but poor information hiding practices may require downstream changes to be communicated to others. Other cases clearly affect others, such as API changes [201].

With the understanding that some technical relationships are "more important" than others, it may be relevant to identify if certain technical relationships must be addressed using specific coordination methods. If there are underlying mechanisms that can identify API changes, for example, these can be flagged as a high-risk change that require specifically in-depth communication. Some technical relationships may be best addressed with awareness through regular meetings or by reading source code instead. The use of the weighted socio-technical congruence measure can represent high-risk changes with appropriately large values that will be considered in the technical relationships matrix. The extensible socio-technical congruence technique described in Section 7.4.5 can be used to separate technical relationships that require explicit communication compared to those that require awareness through the environment only.

7.2.2 The Technical Relationships of Non-developers

A limitation of socio-technical congruence is the lack of involvement of team members that are not involved in the project through explicit task assignments. Team members who are important to team operation, such as environment coordinators and managers should be included in models of coordination for software development. Socio-technical congruence thus far has concentrated mostly on developers and testers because they are easily assigned to tasks.

The problem is that an "assignment" relationship to a work item or to a file is not sufficient to conceptualize someone's work. Ownership, though relevant in some organizations [132], is not universal. In the Ship project, where assignments are made formally through a Microsoft Project plan, the developers are flexible and shift responsibilities frequently to handle incoming issues. The environment coordinators that install and maintain servers and databases were found to be surprisingly central in social network analysis and in email. This suggests that non-developers are extremely important in software development, even if they do not have direct technical association with the software's requirements.

Effort should be taken to extend technical relationships to incorporate dependencies that may involve support staff, managers, and other team members. The relationship of non-developers, such as the environment coordinators, should be investigated in depth to understand their contributions for the project.

One technique to include alternative roles is to include an entity such as a "feature" or "requirement". These entities operate on a higher-level than code does, and may involve testers and managers. Previous work [151] has explored socio-technical congruence as applied to requirements. Similarly, investigating "business requirements" or "models" from model-driven development will involve other types of roles.

A manager is usually responsible for ensuring a set of business requirements are released on time and therefore will probably be connected to everyone. To avoid this type of nebulous assignment, identify dependency relationships between project managers and developers that change based on the progress of the software life cycle.

For example, a technical relationship may connect a manager to an architect during a project's design phase, a developer during the programming phase, and a tester during the project's testing phase.

7.2.3 Consider Weights and Gap Sizes

The weighted socio-technical congruence calculation can be used as a technique for partial alignments of socio-technical congruence. When choosing variables to conceptualize in a socio-technical coordination study, consider how weighted congruence can be useful in the analysis.

The simplest conceptualization that involves weighted congruence is counting the number of technical relationships and social relationships instances, similar to what was done in the RTC study. I have illustrated the use of the gap size measurement in the RTC study. The gap size represents situations where communication may not be sufficient to cover a coordination need. Multiple coordination instances may be required, or a specific kind of medium may be necessary. Incorporating multiple variables for social relationships can increase the weight of a coordination relationship and thus ensure that technical relationships are properly satisfied.

A more elaborate is in the use of an assignment relationship to tasks or projects. An individual is often assigned to a task in proportions, based on the available hours of a work week. Thus, if a person has 40 hours a week, and is assigned 60% to Project 1, and 40% to Project 2, socio-technical congruence can assign this person to 24 hours on Project 1 and 16 hours to Project 2. However, because hours are meaningless in the context of coordinating with others, the technical relationships measurement may be measured in terms of a proportion instead, with a proportion of "coordination time" recommended per person.

Another use would be coordination need relationships having different values depending on the nature of the dependency. For example, a "strong" dependency between two technical entities may be an API change. Such a change could require more attention through coordination than a change that affects internal code only.

7.2.4 The Influence of Emergent People

Emergent communicators in software development have not been studied extensively. An emergent person may represent a previous-unknown source of knowledge or a forgotten coordination need. It may be a new type of user or a forgotten stakeholder. These people may be relevant to coordination patterns and to the project outcome. Managers should evaluate what actions to take when a person emerges through project team interactions.

My research suggests that there are different types of emergent people. In the RTC study, an emergent person was someone who posted a comment on a work item that he was not *technically related to*. In the Ship study, the conceptualization was a person who *was included in a threaded discussion after the first message was sent.*

The core concept is that someone who was not initially involved with a task or a communication was included. In Section 7.3.5, I will propose a model that represents an emergent person as either emerging within or outside of a task, or within and outside of a of a team.

An emergent person can emerge due to changes in technical relationships. Involvement of an emergent people in the technical relationships network (TR) tends to be a large event, because this suggests that someone new was allocated to a project task (the scope of the project was possibly changed), or that there was a change in personnel within the organization. More than likely, an emergent person is involved in the social relationships network SR. It is possible that the presence of an emergent person actually indicates a missing technical dependency or technical assignment in the technical relationships network. When an emergent person appears, that person's link to a technical entity should be identified, if possible. If that person's contribution to the project is significant, then the technical relationships network needs to be updated.

7.3 Refining Social Relationships

The refinement of technical relationships and social relationships occurs in parallel, as was mentioned in Section 7.2. In this section, I discuss conceptualizations of social relationships.

Most conceptualizations of social relationships in related work are represented as explicit communication, usually through electronic means such as email, commenting, or IRC. Other conceptualizations of actual communication, including geographic proximity, have been used [50]. The RTC study reveals that factors other than explicit communication appeared to influence coordination; similarly, the Ship study revealed a number of instances where awareness augmented communication.

7.3.1 Coordination and Awareness

My observations indicate that *awareness inspires coordination among team members*. The word *inspires* is important. Simply being aware of another's actions is not sufficient for coordination. Team members do not communicate explicitly just because they are aware. At the same time, it is not necessary for awareness to exist for communication to occur; for example a message broadcast to an email list requires no awareness of the list's recipients. As a consequence, awareness is neither a necessary, nor a sufficient condition for communication.

Figure 7.2 illustrates the influence of awareness in socio-technical coordination. This figure can be used to guide whether a particular kind of awareness should be modelled explicitly in the socio-technical congruence model, or if it should be modelled implicitly through communication. This decision depends on the availability of data as well as on the type of awareness that is being represented. For instance, if a team is observed to use awareness mechanisms from the environment, then it should be

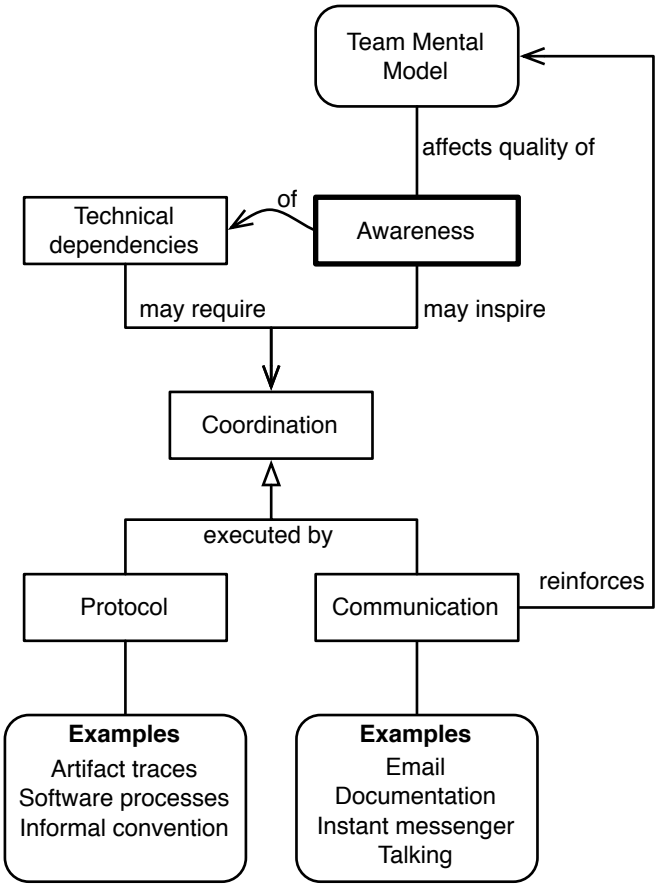


Figure 7.2: The Influence of Awareness on Socio-technical Coordination

presumed that they also have high levels of communication within the team that are not explicitly recorded. Awareness of others' tasks and activities essentially imply that one is cognizant of another's technical dependencies with others and with the environment. The team mental model influences the quality of the awareness that one has of others, and is shaped by communication and interaction with others on the team.

7.3.2 Multiple Variables for Social Relationships

The most common conceptualization of coordination is communication. Communication is an ideal representation of coordination because it is explicit and demonstrates intent. In research, most communication is represented in the form of digital communication, such as email, IRC, instant messenger, issue-tracking–system comments, or similar messages.

Depending on the context of the team being studied, adapting social relationships to use multiple relationships may improve the representation of coordination in the project. Different types of communication may be appropriate for different types of

technical relationships, as discussed in Section 7.2.1. For example, a set of technical relationships involving a group may be determined as being satisfiable through a teleconference meeting, but not through issue-tracking–system comments.

One problem with repository mining is that unrecorded communication probably affects the software-development team, but is not captured. One method to estimate the amount of unrecorded communication within a team is to use awareness relationships instead—a fixed factor can be used as a multiplier to enhance existing coordination instances, for example. Social relationships can be represented as combinations of explicit communication and awareness, on the rationale that awareness inspires coordination. An example would be to "double up" a relationship where an observer has identified two team members as coordinating both in person, and through email.

7.3.3 Including Awareness in Social Relationships

In RTC and Ship, awareness inspiring coordination appeared to occur as a result of three situations: Familiarity with others, experience within the team, and bridging for the team.

One such example of awareness is personal familiarity between team members. In the Ship study, I observed that people were more likely to communicate with familiar team members that they had past project relationships with, and were also willing to cover coordination gaps with those that they have worked closely with. Team members were able to recognize coordination gaps and knowledge gaps among those that they were very familiar with. Task awareness is particularly important: one must be aware of what another person's roles and responsibilities are in order to help that person with specific issues around work. Team feeds in RTC are a way to inform others about a team member's current activities. My findings corroborate Ehrlich and Chang [85], who observed that team members were more likely to seek expertise from people they were familiar with.

Another conceptualization of awareness is reflected by an experienced team member's ability to identify and repair awareness breakdowns. The experienced team members have a strong team mental model and are familiar with their co-workers. In my observations, the experienced team member would identify a potential situation where another team member was not informed; in many instances the other team member was a "junior" person that the experienced team member was a mentor for. This is most evident in the situation involving the current test leader and the requirements change—the person who informed the test team leader was not on the project, but had served as a mentor and a test leader in the past. Our email data supports the fact that the test leader and the test leader's mentor were very familiar with each other's work. Both team familiarity and task familiarity give a team member the ability to recognize when others may not have the knowledge needed to make proper decisions in the project. This strength of familiarity is, in a way, a method for team members to generate "congruence" on behalf of their team mates.

Finally, team members should be given special consideration as potential bridges of actual communication. Previous work studying brokerage in software engineering considers information as a "flow" [151], but while this view is simple to model, some have contested this perspective as overly simplistic [17]. One way to avoid thinking in terms of flow is to identify who in the network is representing the interests of another person in the network. When investigating the case, identify if the experienced team member acts as a broker on behalf of others, whether indirectly or directly, and connect these two people. For instance, Nancy in the Ship study was clearly acting in the interests of Lilly when sending her information about requirement changes, but without interrupting the rest of the project. If this context can be captured in the network, then it may be possible to identify not only sources of potential awareness breakdowns, but also political posturing too, in the case that one person acts in the interests of others to the detriment of the project.

Thus, when selecting variables to represent social relationships in a socio-technical environment, consider the possible awareness channels that the team members have with each other, especially through the environment. Use multiple relationships representing both explicit communication and awareness if necessary.

7.3.4 Considering Weights and Gap Sizes

Weighted congruence can be used in actual communication relationships similarly to how it can be used for coordination need relationships described in Section 7.2.3.

The use of proportions can be assigned to social relationships in lieu of precise instances of coordination. A developer does not usually track the number of instances of communication with another person, but may be able to estimate the amount of time spent communicating with that person using different communication media. A developer then may be able to claim that 40% of the time communicating with this other person was over email, 10% was in meetings, and 30% was through instant messenger. These percentages may be used when evaluating communication media for satisfying certain types of technical relationships.

7.3.5 Modelling Emergent People

I discuss a conceptualization of emergent people below that considers emergence from both the technical perspective as well as the social perspective. The people who emerge are considered stakeholders in the project. This conceptualization involves different classes of emergent stakeholders and suggests they can be handled when interpreting their appearance in socio-technical coordination.

In socio-technical coordination, there are two entities that are "emergent": *emergent people*, and *emergent interactions*. An emergent person is a person who was previously not known about, but was included in a discussion. An emergent interaction is an interaction that is not expected—that is, not an interaction that was

identified by the technical relationships relationship between two people. By definition, any interactions with an emergent person are emergent interactions.

I suggest two attributes to describe the context in which people emerge.

- Task emergence: People are emergent because they are not attached to a task, but became involved in that task
- Boundary emergence: People are emergent because they are outside of the boundary of a formally-defined team

I summarize this description in Table 7.1. There are four cases that can be identified.

		Boundary emergence	
		Within team	Emergence outside team
Task emergence	Within task	Team members that are familiar; no emergence	Interaction across boundaries, possibly with the assistance of a point-of-contact
	Emergence outside task	Team member providing expertise or sharing awareness	Possible stakeholder making new requests or reporting issues

Table 7.1: Table Describing Variables Affecting Emergence

Within task; within team. This situation describes the case where there is no emergence. The individuals are assigned within the task, and belong to the same team as each other. This should generally be considered a nominal case, since it is not necessarily surprising for a team member to contribute to a task that is technically challenging.

Emergence outside task, within team. This situation describes the case where a team member assigned to a task consults with a team member who is not assigned to that task. This situation usually illustrates some kind of expertise-seeking or awareness-sharing, though it may also illustrate a previously-unidentified dependency.

For example, two developers are coordinating about an issue, but may require some assistance from a senior developer. They see the senior developer, get advice about how to handle the issue, and then continue on their way. Certainly, the senior developer does not have a technical dependency on these two developers, but the two developers were in need of coordination from the senior developer. This particular example illustrates a situation where there is emergence within the team. The senior developer contributed to the tasks of two

developers, but at the same time, the coordination need is not essential—simply helpful.

Within task, emergence outside team. Someone is assigned to the task but is outside of the team. In this situation, a point-of-contact acting as a broker may be present, as a broker monitors information into and out of the team [151]. If the point-of-contact is not known in advance, then some actions may be recommended to ensure that the team members are comfortable with this additional person.

Outside task, emergence outside team. Someone emerges from outside of the task, and is outside of the immediate team. In this case, the emergent person may be a stakeholder making requests. There may be an undetected technical dependency in the project. If the expertise must come from someone outside of the team, there may be an organizational gap.

One question is whether technical relationships need to be modified based on emergent people. This could be accomplished either by a manager explicitly modifying a project plan to include the contributing emergent person, or by having a socio-technical congruence system extract these implicit technical contributions automatically. At this time I do not believe that technical relationships need to be modified to involve emergent people, and rather to keep an emergent person "on the mind" based on previous contributions. Identifying and recording emergent people would be one way to design expertise recommendation tools that handle situations especially that involve experts outside of the team.

7.4 A Process To Guide the Application of the Socio-technical Congruence Model to Study Socio-technical Coordination

The model can be applied by using stepwise process described below, and visualized in Figure 7.3.

1. Select the study focus. *Identify the response variable.*

2. Conceptualize technical entities; also consider interactions in technical relationships variables, weights, and emergent people. *Apply the conceptualizations from the model at this point. Consider the alignment techniques from the model.*

3. Conceptualize social relationships; also consider interactions in social relationships variables, weights, and emergent people. *Apply the conceptualizations from the model at this point. Consider the alignment techniques from the model.*

4. Gather data.

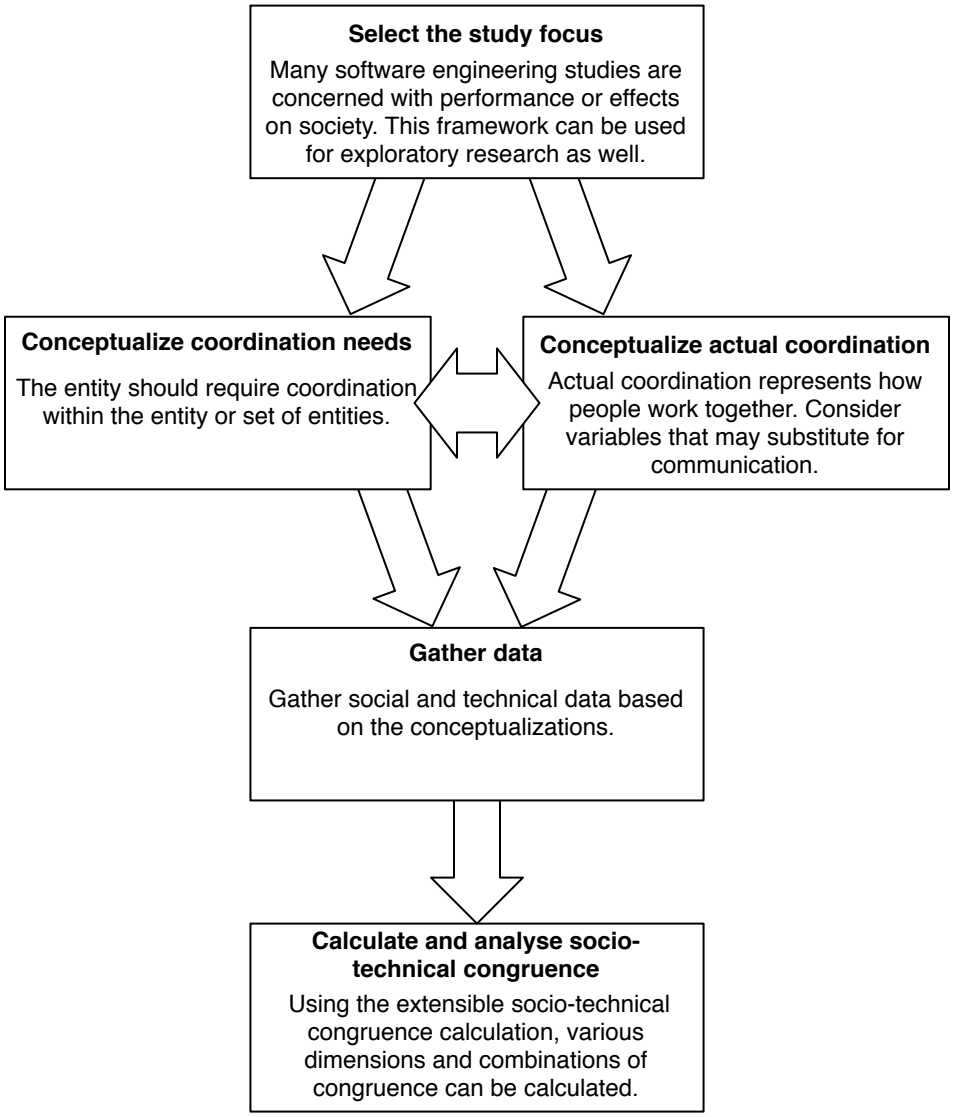


Figure 7.3: A Process to Apply the Socio-technical Coordination Framework

5. Calculate and analyze socio-technical congruence. *Apply alignment techniques from the model. Examine situations from the model at this point.*

7.4.1 Select the Study Focus

The first question is to determine what phenomenon in human-centric software engineering to study. The objective is to discover if there is a relationship between a number of treatment variables, one of which is socio-technical coordination, and a response variable.

Select a focus that is likely to be affected by coordination and technical work. This tends to rules out research around a single developer, as no coordination is required

in a single-developer case. In addition, extremely small groups of four or less that do not have external dependencies are not likely to benefit from the socio-technical coordination model because such group sizes tend to be trivial to manage.

Most software engineering research studies what affects the performance of the software development process, often with the intention of improving the performance. Some examples of response variables are as follows.

- **Software cost.** As software is labour-intensive, the time required to build software directly translates to cost. One common conceptualization is the defect resolution time [50, 166]. Another one can be the return-on-investment of a software project.
- **Software quality.** Quality is often conceptualized as the number of bugs [107] or as the number of post-release failures [26].
- **Societal impact.** Factors such as the number of users or the satisfaction of developers may also be considered as response variables.

This is by no means a complete list of variables that can be studied using the socio-technical coordination model. These outcomes tend to be influenced heavily by the work practices of a software-development team, and as a consequence, investigating socio-technical coordination is relevant.

Another focus could be an exploratory study that examines a specific issue. For example, one may examine the relationship between two different functional teams such as support and engineering; the effects of a new office configuration on software teams; the influence of multiple stakeholders on a software team; or even the effects of free lunches given to developers on software architecture. These issues fundamentally involve interactions between people, their work, and their work environment.

7.4.2 Conceptualize Technical Relationships

Using the considerations presented in Section 7.2, one can conceptualize, model, and begin the process of understanding the relevant technical relationships within an organization. This step in the process should be done in parallel with conceptualizing social relationships, as in most organizations, the technical aspects and coordination aspects are highly interrelated.

When finalizing the conceptualizations, complete the socio-technical congruence template described in Section 3.8.

7.4.3 Conceptualize Social Relationships

As above, one can use the considerations in Section 7.3 to conceptualize and understand social relationships. This step can be done in parallel with conceptualizing technical relationships.

7.4.4 Gather Data

When the conceptualizations of the social and the technical variables have been decided, the researcher gathers relevant data for study. Discussing in detail the data collection step of the research is again beyond the scope of this research, and other resources on general research design and data gathering are more appropriate here (ex: [227, 170, 61]). However, I will discuss some conceptualizations of technical entities, technical relationships, and social relationships that may aid data collection.

The techniques inspired by the model allows a synthesis of data from multiple data sources. While contextual, qualitative data can be collected, it is necessary to collect some kind of repository data or project documentation in order to properly model technical dependencies and derive technical relationships.

To collect data for technical relationships, use repository data, particularly of the source code versioning system. This will allow a researcher to gather technical data that includes the links between people and artifacts. It is possible to gather technical data based on documents as well, though this requires a significant amount of manual work on behalf of the researcher. Technical documentation for a project will tend to be focused on features or requirements, and are therefore higher-level than source code modules.

To collect social relationship data, consider two approaches. First, collect instances of explicit communication. If possible, electronic repositories should be collected. Electronic communication media that is automatically recorded, such as work item comments, emails, and chat logs are excellent sources of communication. If not, then a questionnaire that asks "Who do you communicate with about [task]?" and "Who are you comfortable working with on [task]?" may be a sufficient substitute.

Collect knowledge of the social interactions among team members to learn about their awareness patterns. Observations will reveal relationships within the workplace: In particular, look for unscripted interactions between team members, such as others who visit their desk. Interviews about the context of their work and how they get information about others are also important. Questionnaires can identify which team members communicate about what topics to each other. Pay particular attention to gathering a team member's experience in a role as well as the amount experience working with each other person on the team, as it appears to govern their ability to be aware of tasks in their socio-technical environment.

There must be a link between the social relationships and the technical relationships. This link is conceptualized by the connecting entity (Section 3.5.3).

The most important aspect of the data collection step is to ensure that there is a reliable method to acquire both technical and social data. Both are required to perform effective analysis on socio-technical coordination and socio-technical congruence cannot be calculated if one of these dimensions is missing. There are many techniques for gathering this data, such as the Echo method [178] that are beyond the scope of this dissertation.

7.4.5 Calculate and analyze Socio-technical Congruence

The socio-technical coordination model invites a researcher to incorporate multiple relationships that represent communication, coordination, and technical dependencies. Once socio-technical congruence has been calculated, congruence, as a variable, can be used to determine its effect on the study focus. While the socio-technical congruence value is a convenient number to use for statistical models, keep in mind that examining the networks themselves, particularly the lack-of-coordination network, can reveal much about how individuals collaborate about technical work.

A limitation with the existing congruence calculation is that incorporating multiple relationships is not well-defined. Existing work has calculated different congruence measurements for each variable, such as "geographical congruence" or "modification request congruence" as presented by Cataldo et al. [50], but no technique combines these variables in such a way that some variables compensate for others. For instance, Cataldo et al. [50] use four types of social relationships in their study but considers each type independently.

In this section, I present a technique that can be used to incorporate multiple relationship types into the socio-technical congruence calculation. This multi-relationship technique allows the researcher to investigate congruence especially in a situation when different social relationships interactions can satisfy technical relationships. An **aggregated congruence measurement** incorporates multiple relationships and considers them as a grouped relationship that can satisfy technical relationships. This is important if there are multiple communication media that do not need to overlap as actual communication. For example, if it is sufficient for a software developer to communicate in either IRC, or in instant messenger with another developer to fulfill a coordination need, then the IRC relationship matrix and the instant messenger relationship matrix can be multiplied together to identify the relationship. This extensible calculation can also be used in conjunction with the weighted congruence measure described earlier in this thesis (Section 3.6.2).

The Multi-variable Socio-technical Congruence Technique

This technique combines a number of attribute matrices to calculate either the TR or the SR matrix. Any number of extra "attribute" matrices could be created. For example, to handle the issue of transitivity in socio-technical congruence, where "A talks to B, who talks to C, therefore A and C have sent information to each other", I could generate a transitivity matrix TN that highlights transitive relationships only.

Thus, any variable can be inserted into general $n \times n$ relationship matrix M, where n is the number of people in the organization. The value of m_{ij} indicates the relationship strength between person i and person j.

An attribute can also be derived using an intermediate matrix, similar to Equation 3.1. For example, role relationships may influence interactions in the organization. A person is assigned to a role, and roles are allowed to interact with other roles. The result of the calculation indicates who is permitted to communicate with whom

when their roles are considered.

To calculate an $n \times n$ coordination matrix C, where n is the number of people in the organization, first identify a $n \times k$ assignment matrix A, where n is the number of people in the organization and k is the number of attribute values to be assigned. Then, identify a $k \times k$ dependency matrix D among the attributes. If k_{ij} is 1, then the element at i and the element at j have some kind of relationship with each other.

The final calculation of the matrix C is as follows.

$$C = A \times D \times A^T \tag{7.1}$$

The resulting matrix C indicates the people who have a relationship with each other, assuming that the attributes that they possess also have a dependency as indicated in D.

Examples of applying these attributes will appear below in an example (Section 7.5.1).

Aggregated Congruence Measurements Incorporating Multiple Variables

Any number of additional relationship networks can be used to conceptualize either social relationships, or technical relationships. An **aggregated congruence measurement** describes the situation when multiple relationship matrices are multiplied together.

The matrix algebra presented here is similar to the formula from previous socio-technical congruence studies [50, 139]. However, rather than specifying that every technical relationship can be satisfied by every social relationship, this aggregated technique can be used to "select" specific individual technical relationships that should be satisfied by a specific type of social relationship.

This selection allows for the following benefits. First, one can calculate the alignment for a specific kind of relationship—for example, one may find that meetings align well to technical relationships, but that written forms of communication like email and issue-tracking descriptions do not. Second, relationship data can be stored independent of other types of data, making data management easier.

Any variable can be used in an $n \times n$ relationship matrix M, where n is the number of people in the organization. The value of m_{ij} indicates the relationship strength between person i and person j.

Either a technical relationship network or a social relationship network can be modified using an intermediate attribute. The following equations are best explained with an example situation.

In a software team, we can calculate an initial set of technical relationships TR may be calculated by multiplying person-to-file assignments TA, file-to-file dependencies TD, and the transpose of TA.

$$T = TA \times TD \times TA^t \tag{7.2}$$

The resulting network indicates which individuals should be coordinating. This calculation is the one that can identifies the relationships a network such as the one shown in Figure 1.1(c) under Section 3.6.1.

Once the technical relationships TR are calculated, we can "remove" links that are irrelevant or satisfied by social relationships. Thus, if you have a matrix S that identifies technical relationships that are not important, you can subtract them from the original TR matrix as follows.

$$t'_{ij} = t_{ij} - s_{ij} \text{ for } i = 1, \ldots, m \text{ and } j = 1, \ldots, m \tag{7.3}$$

What an appropriate S matrix depend on the context of the case being studied, and the researcher will have to identify what technical relationships may not be relevant in a particular project. For example, there may be a relationship between two people that is known to be covered because these two people work in the same office. One can then identify in the S matrix that these two individuals are aware of each other's technical relationships, and that the particular link is satisfied.

We can calculate congruence using the modified T'. Because links are removed, congruence values will increase, but this technique allows the researcher to essentially remove exceptions in a technical network.

Other operations can be substituted for "subtraction" in Equation 7.3, including boolean OR ($\vee$) and boolean AND ($\wedge$). Figure 7.9(d) for instance would be the combination of a "brokerage" matrix and an "instant messenger" matrix using a boolean OR relationship.

In order to calculate the matrix for Figure 7.9(d), simply take a binary OR relationship between the corresponding elements of the two social relationship matrices in a scalar fashion. Thus, given the transitive-social relationship matrix TS and the communication-social relationship matrix CR we would calculate each entry in an aggregated social relationship matrix SR':

$$sr'_{ij} = ts_{ij} \vee cr_{ij} \text{ for } i = 1, \ldots, m \text{ and } j = 1, \ldots, m \tag{7.4}$$

Multiple relationships can be calculated using different matrices. Instead of a transitivity matrix S, one may use a co-location matrix C instead and calculate T'' instead. Multiple relationships represnting different types of social relationships, as well as different types of technical relationships can be devised. Thus, the general form of Equation 7.1 and Equation 7.3 can be used to calculate both a modified set of social relationships and a modified set of technical relationships.

The final congruence index can be calculated as a ratio of the number of connections in SR' over the number of connections in TR', as in the original congruence calculation.

By combining attribute matrices in different ways to create TR and SR matrices that are eventually used in the final congruence calculations, a researcher can build different areas of investigation to study the relationship between people in software development. This flexible extension allows socio-technical congruence to be calculated for a large number of situations.

7.5 Example Applications of the Socio-technical Coordination Model

In this section, I describe examples of how to conceptualize socio-technical coordination in various cases, with a particular emphasis on the application of the multiple-variable congruence technique.

7.5.1 Example 1: A Role-based Congruence Measurement

The context of the empirical investigation determines the construction of relevant relationship matrices. As mentioned above, a relationship matrix can be combined with other relationships to form a modified SR matrix or TR matrix. This example is adapted from an empirical investigation by Marczak [148]. An conceptualization appeared in Marczak et al. [152], and the work was further developed into the exploration of a role-based measurement [149].

In a team called App, the organizational structure of the team mandated that certain roles should not communicate directly in order to distribute information throughout the team. This rule was put into place to prevent developers and testers from making crucial decisions without input from senior team members and management. For instance, developers could communicate with each other, developers were not allowed to communicate with testers directly. A diagram of the roles in this project, along with their permitted connections, appears below in Figure 7.5.

The question is, what influence does this prescribed communication structure have on the team? Do the team members follow this structure, or do they extensively communicate with each other despite the rule? If the communication structure is not followed, this may suggest that the team must engage in communication across roles to coordinate their work. The socio-technical congruence template for this example appears in Figure 7.4.

To calculate the role-based congruence measurement RC, combine a role-assignment matrix RN and a role-dependency matrix RD as in equation 7.1. The result is

$$RC = RN \times RD \times RN^t$$

Visually, we can represent this as a network such as Figure 7.6. In this example, the arrows indicate the assignment of people to roles. RA is a requirements analyst, Dev Lead is a development leader, Dev is a developer, and Tester is a tester. The thick lines represent the role dependencies that dictate that those roles should discuss their tasks. A requirements analyst interacts with development leaders, development leaders interact with developers. Roles can interact with others of the same role.

The resulting role-based coordination network looks like Figure 7.7(a). Because the testing role does not have a specific link to the other roles, he is isolated. This role coordination network illustrates the roles of people in this team that should coordinate regularly. The precise meaning of the role coordination matrix depends on the case under investigation. In this organization, it is a list of people *who should*

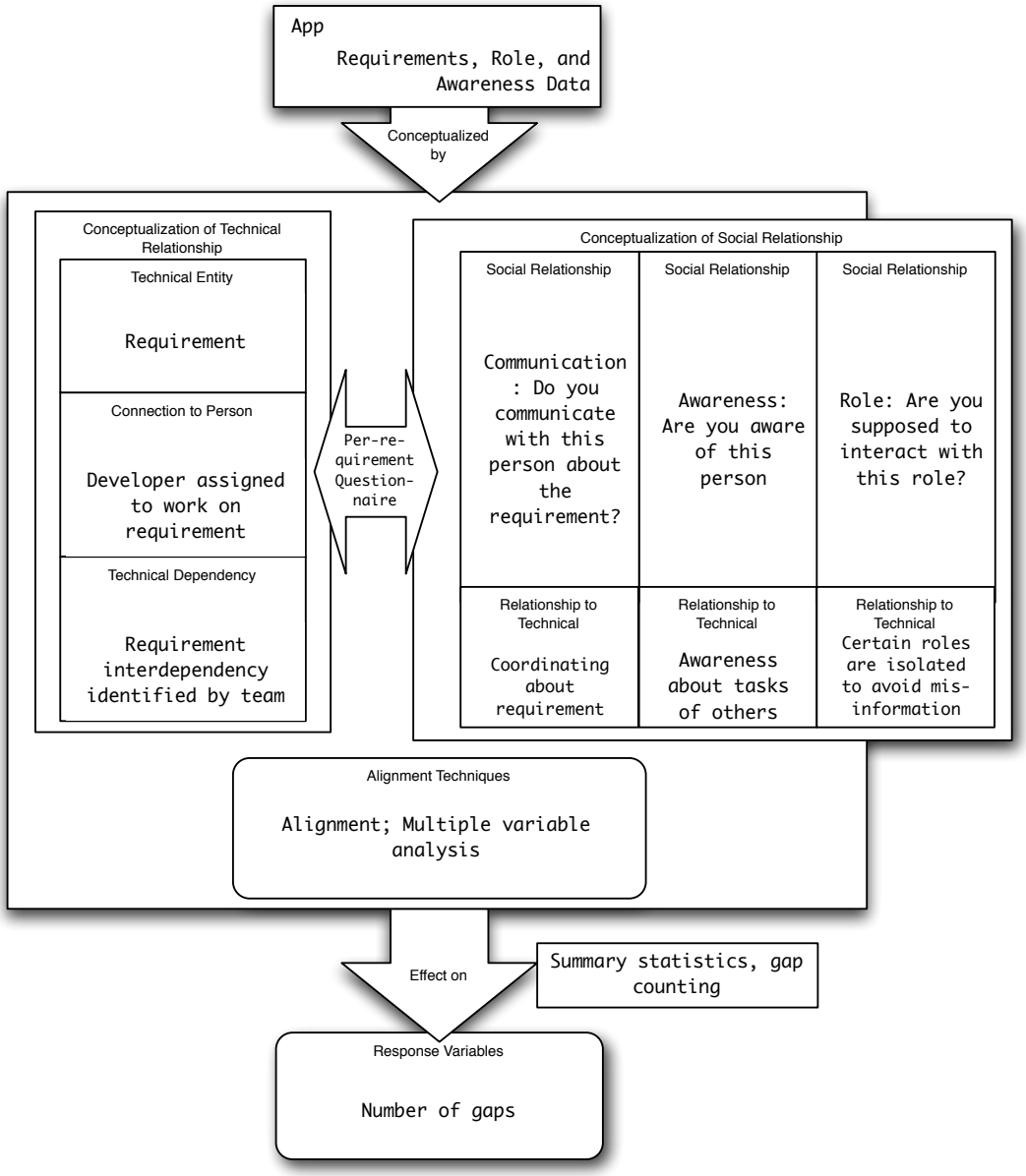


Figure 7.4: Socio-technical Congruence Template As Applied to Marczak et al. [152]

communicate and *who should not communicate*, based on their role. However, in another organization, this may be a recommended communication. It may also indicate communication that happens every regularly (perhaps there is a SCRUM meeting that involves everyone except the tester).

We can then combine the *RC* matrix with a "technical relationships" matrix calculated normally. In our case, we apply this to a social relationships matrix to come up with a modified set of technical relationships (Figure 7.7(b)). Let us assume that, because this structure indicates who should not explicitly communicate with

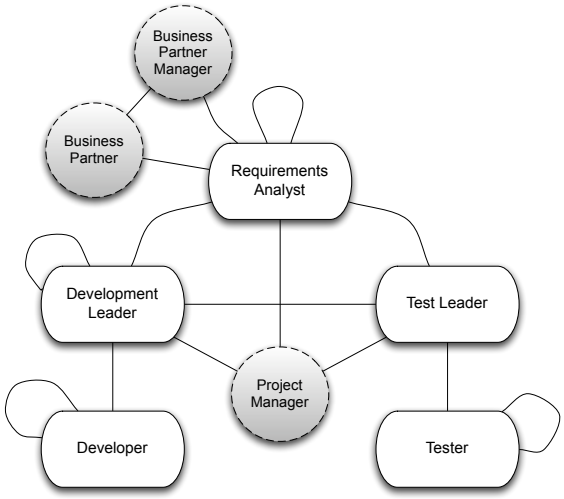


Figure 7.5: Permitted Role Interactions in App Software Team

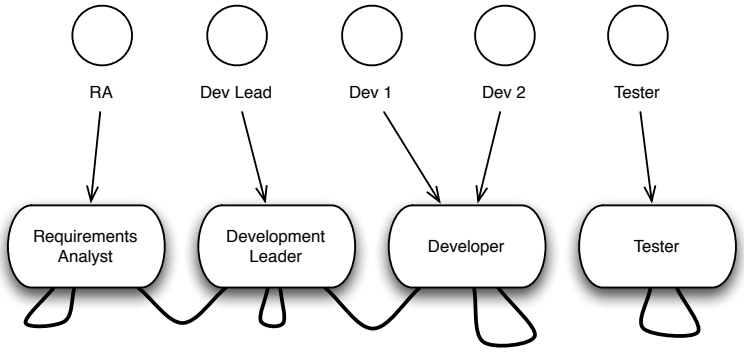


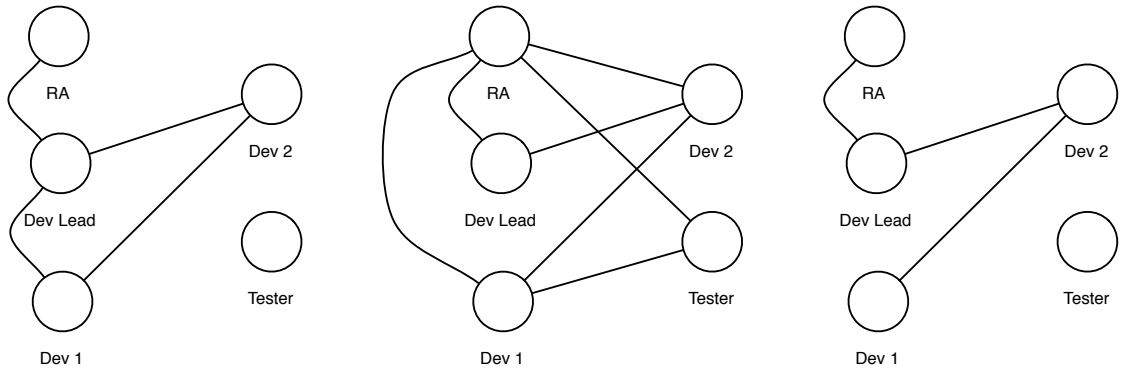
Figure 7.6: Creating the Role Assignment and Role Dependency Networks

others, that we incorporate it into the modified TR' matrix. The calculation therefore appears as follows. The resulting network is shown in Figure 7.7(c).

$$cn'_{ij} = rc_{ij} \wedge cn_{ij} \text{ for } i = 1, \ldots, m \text{ and } j = 1, \ldots, m$$

The TR' matrix and the SR matrix can be overlapped to calculate the congruence index.

The extensible congruence measurement allows a researcher to conceptualize different types of "communication" and "gaps". For example, if the actual communication is represented by dashed lines in Figure 7.8, there are some aligned communication instances and some unaligned communication instances. The communication between RA and Dev Lead, and Dev Lead and Dev 2 are "aligned" because they correspond with the TR' relationship. In contrast, there is no communication between Dev Lead and Dev 1. This communication is role-based because they do not have a coordination need from technical dependencies, but are connected in the role



(a) The Role Coordination Network, RC.

(b) The Technical Relationships Network, TR.

(c) The Combined Role and Technical Relationships Networks, TR'.

Figure 7.7: Combining Technical Relationships and Role Relationships to Create a New Technical Relationships Network

dependency. There is also communication between Dev 1 and Tester. This is communication that is not required by either their role relationship or their technical dependency, and therefore this may be seen as "backchannel communication". The gap between Dev 1 and Dev 2 is a "real gap" because they should be communicating, but do not do so. For an in-depth explanation of the meanings and implications of these relationships, please see Marczak et al. [149].

The identification of these different types of gaps allows for an in-depth examination of the relationships between individuals and provides more information about how they work than using the simple congruence measurement. This allows an analyst to examine work assignments, organizational structure, and impacts of awareness tools on a team.

7.5.2 Example 2: Representing a Point of Contact

Software development involving multiple sites and dependent teams can cause problems with over-coordination. To alleviate this, many teams use a point of contact in another team toward which to direct communication. The point of contact acts as a broker for incoming information and forwards the information accordingly to internal team members. The existing congruence gap may identify a necessary connection going from a team member to internal team members of the other team when instead they should be speaking with the point of contact. The question is, is a transitive connection with a Point of Contact person A sufficient to ensure coordination between Person B and Person C?

I illustrate how such a study can be conducted in Figure 7.9. For example, we have three developers, and C depends on A for communication (Figure 7.9(a)). In this particular situation, B is an experienced team member, and communicates

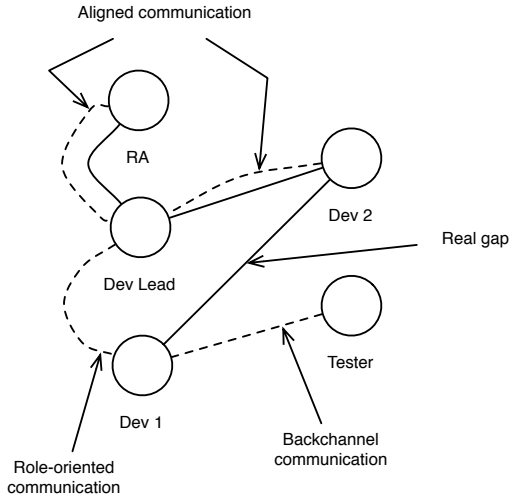


Figure 7.8: The Actual Communication Network SR on Top of the TR' Network.

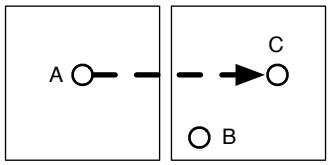
information from A to C. Because we know that this brokerage occurs, we can assume a transitive-social relationship from A to C (dotted line in Figure 7.9(c)). Because we know that B acts as a broker, this transitive relationship can serve in place of a direct communication-social relationship between A and C (Figure 7.9(b)). Either Figure 7.9(c) or Figure 7.9(b) satisfy the relationship in Figure 7.9(a).

We can combine both social relationship networks into Figure 7.9(d). In this particular example, either Figure 7.9(b) and Figure 7.9(c) may be considered as "satisfying" the technical relationship.

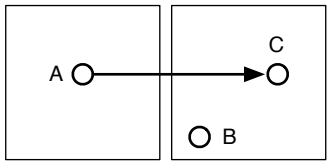
This technique combines a number of attribute matrices to calculate either the technical relationships matrix TR or the social relationships SR matrix. Any number of extra "attribute" matrices could be created. For example, to handle the issue of transitivity in socio-technical congruence, where "A talks to B, who talks to C, therefore A and C have sent information to each other", we could generate a transitivity matrix TN that highlights transitive relationships only.

In this example, the transitivity relationship is conditional on the fact that B $\longleftrightarrow$ A and A $\longleftrightarrow$ C communicate. The calculation of the transitivity network based on the communication relationship matrix is relatively straightforward. This extension to the socio-technical congruence model allows these kinds of operations to be modelled in a modular manner.

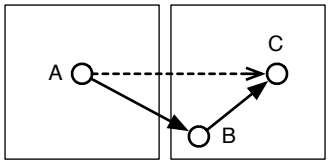
The transitivity relationship used here could be combined with the role relationship explained above in Section 7.5.1 to explore the effects of transitivity and roles on social relationships and technical relationships. It is possible to combine multiple relationships *ad infinitum*. In this sense, the model allows for a significant exploration of social relationships and their alignment to the technical relationships among team members.



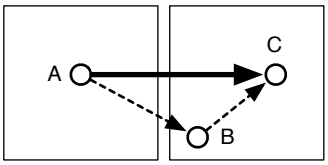
(a) Technical Relationship: A Should Send Information to C



(b) Communication-Social Relationship: A Communicates with C Through Instant Messenger



(c) Transitive-Social Relationship: A is Connected to C via Experienced Developer B



(d) Aggregated Social Relationship: Aggregated Link Combines Both Brokerage and Communication

Figure 7.9: Technical and Social Relationships for a Point of Contact

7.5.3 Example 3: Multi-variable Relationship Modelling of an Organization

In this example, I present a theoretical example of aggregated congruence measurements that include meetings. The purpose of this example is to demonstrates how the model uses multiple technical and social relationships.

In an imaginary team, we have one development leader and three developers working on three components (Figure 7.10). The Dev lead and Developers 1 and 2 are on the same geographical site. Component B depends on Component A, and the three components each integrate into a library. The developers communicate using meetings and email. The focus of this investigation is to identify which developers need to be kept informed of changes to the software library.

In this case, some information is missing: we do not know who is responsible for the library. Thus, an adjustment must be made to the task assignment network. We assume that the relationship from the developers to the components also extends to the library itself, and therefore every team member has a small part in ensuring that the library works. A new technical relationships network can be created for the library itself. Now, there are two technical relationships networks: one for the components (Figure 7.11(a)), and one for the library (Figure 7.11(b)).

The technical relationships network is illustrated in Figure 7.11(c). The bold lines indicate a "level 2" dependency, where there is both a component coordination need and a library coordination need.

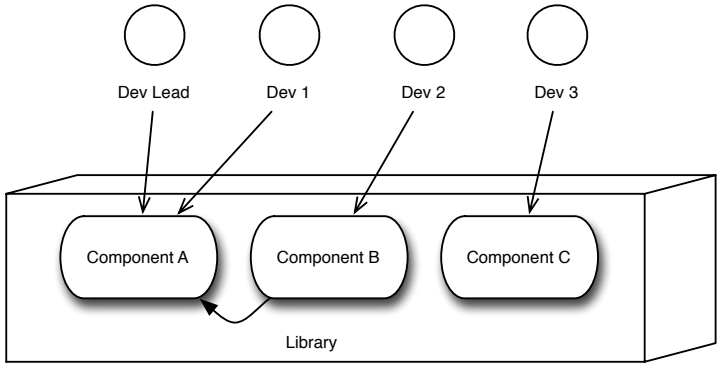


Figure 7.10: The Task Assignment and Technical Dependency Networks for a Sample Project. The Library Encompasses the Three Components.

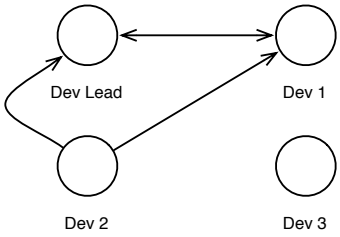
These developers communicate in two ways: regular meetings, and emails. Some of the developers meet regularly; we can model this as one relationship in an social relationships network (Figure 7.11(d)). Because the meeting involves three of the developers, we can presume that the connection is bi-directional. We model an email connection between developers if they communicate above a certain frequency per week (Figure 7.11(e)).

Once we model each of the communications, we can combine them together into a complete picture of social relationships. In this case, we simply overlap the various communication media. The complete social relationships network in Figure 7.11(f). If desired, we could have specified weights to the relationships to indicate, for example, that a meeting is "twice as effective" as an email for the purpose of satisfying technical relationships.

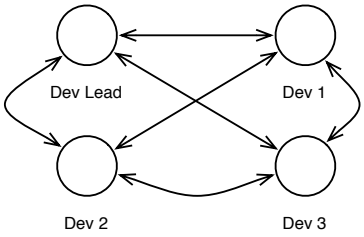
Already, we can see a number of trends. First, Dev 3 is kept in the loop through email, but is not involved in meetings. Dev 2 and Dev 1 do not send emails to the Dev Lead frequently enough for their communications to appear in Figure 7.11(e). A plausible explanation is that Dev 1 and Dev 2 feel adequately informed through meetings and therefore do not communicate with the Dev Lead or with each other.

The meeting that involves the Dev Lead, Dev 1, and Dev 2 (Figure 7.11(d)) aligns perfectly with the technical relationships network for the components (Figure 7.11(a)). Dev 3 is kept in the loop through e-mail from the Dev Lead—perhaps it is the Dev Lead's responsibility to ensure that the off-site Dev 3 is kept informed of the results of their meetings.

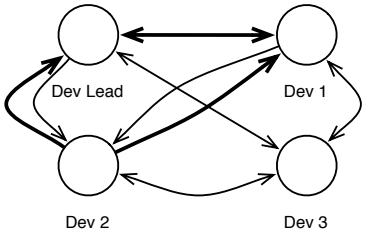
The lack-of-communication network that illustrates the gaps is applicable only to Dev 3's involvement with the library (Figure 7.11(g)). While there are gaps in this network, it may be the case that Dev 3 is not actually involved as much with the library as we may have assumed. If this is in fact the case, the technical relationships network for the Library (Figure 7.11(b)) could be adjusted in the future to explicitly exclude Dev 3. In the future, if the developers adopt a new tool, a new social relationship network can be calculated and incorporated into existing social relationship



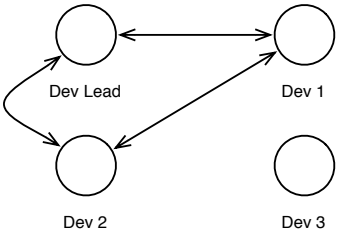
(a) Technical Relationships Network for the Components



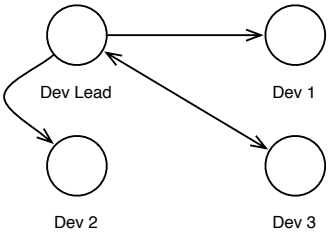
(b) Technical Relationships Network for the Library



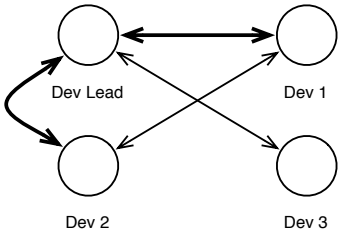
(c) Aggregated Technical Relationships Network



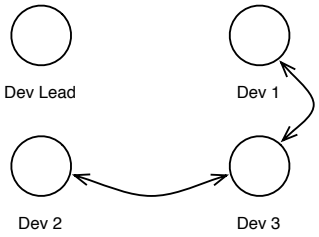
(d) Meeting Network



(e) Email Communication Network



(f) Aggregated Social Relationships Network



(g) Lack-of-Communication Network

Figure 7.11: Technical Relationships and Social Relationships for the Meeting Example

networks while preserving the integrity of previously-gathered data. In this respect, this socio-technical congruence model is extensible.

7.6 Differences from Cataldo's Socio-technical Congruence Research

The socio-technical congruence model in this dissertation builds on Cataldo's socio-technical congruence research [44, 48]. Cataldo in his dissertation describes a "framework for identification of work dependencies" (Chapter 2, p.13). In particular, that chapter examined a relationship between a developer and technical work, with an emphasis on using the results for coordination. The stated objective of his dissertation is "... [to] provide a fine-grain level of analysis of coordination" and "to allow for identification of work dependencies from alternative representations of technical dependencies of the system". Cataldo's dissertation has a heavy emphasis on work dependencies and the technical aspects of software development. As discussed in Section 3.10, there is a wide body of knowledge that includes Cataldo's work that is devoted to examining technical dependencies. While socio-technical congruence in Cataldo's dissertation focuses primarily on the identification of work dependencies within a software system, my dissertation focuses on the conceptualization of social relationships within a software team.

The socio-technical congruence model goes beyond the descriptions provided in Cataldo's dissertation by describing elements that need to be considered when designing studies that use socio-technical congruence as well as expanding and refining some of the conceptualizations that should be used for social and technical relationships. It also includes a guideline on how to apply socio-technical congruence to the study of socio-technical coordination.

7.7 Limitations of the Model

The socio-technical congruence model is a more complete representation of the current state of socio-technical congruence. However, there are still a number of limitations with the model.

Socio-technical congruence uses a number of conceptualizations to estimate socio-technical coordination. In the RTC study, I have identified empirical evidence regarding socio-technical congruence's relationship to team performance using the conceptualizations of technical relationships based on co-changed files, and social relationships as comments. My study has revealed that socio-technical congruence is beneficial in certain situations, but more studies need to be done in order to generalize this result to software organizations in general. The contribution that RTC provides is that there is a situation in which high socio-technical congruence does not correlate with successful integration build results. This may be a consequence of an overload

effect, where it is too difficult to expect each team member to coordinate their efforts based on dependencies. The RTC empirical investigation revealed that congruence might have an "upper bound" of participants before it begins to break down because it recommends too many necessary social relationships. However, without further analysis this cannot be conclusive.

The guidelines are not a single, true approach to creating a socio-technical congruence study. Thus, while many of the guidelines may help guide a researcher who wishes to apply socio-technical congruence, they are still not at the level of rigour that would constitute a process. More research is required to be able to completely define a process describing the application of socio-technical congruence.

There is also no concrete advice for handling long-term longitudinal studies in the socio-technical congruence model. The model currently can be applied to snapshots, or may be applied to regression models. One way to handle longitudinal studies would simply to be to examine multiple snapshots taken over time. Some prior work has attempted to address the issue [216] of time, but more exploration of such techniques would be beneficial.

The model is a preliminary work that runs short on actually stating what is "good" for a software team, and concentrates primarily on exploration instead. Achieving good performance is influenced by too many other factors for the model to recommend specific practices. More empirical studies, as well as theoretical models, would benefit the study of congruence and move it toward a state where it may (or may not be) a recommended metric for software team performance.

Chapter 8

Contributions and Future Work

Socio-technical coordination refers to coordination that is performed in an environment with social and technical factors, such as a global software-development team. A promising technique that captures and measure socio-technical coordination is called socio-technical congruence, but the measurement has a number of limitations in its conceptualizations, including **binary gaps**, **difficulty comparing socio-technical congruence studies**, **unknown influence of awareness**, and **not including relevant team members**. As a result, the goal of this thesis was to develop a model of socio-technical congruence used for the study of socio-technical coordination. The model describes refined conceptualizations of social relationships, improves techniques to calculate alignment, and describes an approach for applying the model. Such a model allows a researcher or a practitioner to examine socio-technical coordination from a high-level perspective.

Thus, the research objectives were as follows:

> **Objective 1: To develop a socio-technical congruence model that explains what conceptualizations belong in the model, what elements belong in the model, and how to calculate alignment.**

I developed this model by examining related work, but identified that there were still outstanding limitations that needed to be addressed by conducting empirical studies.

> **Objective 2: To examine awareness and to identify how awareness within a global software-development team can be conceptualized within a socio-technical congruence model.**

I performed an empirical investigation of awareness in the Ship project and identified the need for multi-variable relationship modelling and refined social and technical relationships in order to reflect how team members use communication in parallel as well as experience and brokerage to fill gaps for others.

Objective 3: To examine communication patterns, which includes important communicators and emergent people, and to identify how communication patterns can be represented within a socio-technical congruence model.

I performed an empirical investigation of communication patterns in the Ship project and identified that technical relationships may have to be conceptualized so that managers and supporting staff are included in socio-technical congruence networks. In addition, the presence of emergent people suggested that technical relationships may have to be modified over time to reflect the involvement of these new people.

Objective 4: To apply the socio-technical congruence model to determine the relationship between socio-technical congruence and the performance of a global software-development team working on coordination-intensive activities.

I applied the socio-technical congruence model to study the relationship between socio-technical congruence and team performance. I found that high socio-technical congruence can benefit builds, but only in certain situations, and that high socio-technical congruence actually reduced performance. I also identified that team members appeared to coordinate more closely when problem builds occurred. These findings illustrated the value of the socio-technical congruence model as well as the value of the gap size technique.

Objective 5: To improve the socio-technical congruence model developed in Objective 1 by incorporating the results of the empirical investigations from Objectives 2, 3, and 4.

I revised the model based on the empirical findings of the studies above. I described guidelines to conceptualize the entities in socio-technical congruence and described an approach that aids a researcher in applying the socio-technical congruence model to study socio-technical coordination.

8.1 Contributions

The contributions of this research are two-fold. First, I presented a socio-technical coordination model that can be used by investigators to construct research around software engineering teams or organizations. Second, I presented a number of empirical findings from empirical investigations.

Below, I outline the empirical findings, then summarize the model in light of these findings.

8.1.1 Empirical Findings

Team Members Communicated Using Communication Media in Parallel

Team members in Ship used multiple forms of communication to stay aware, including phone meetings, emails, instant messenger, and personal visits. I also observed in RTC that developers also use the integrated development environment to study traces left by others on the environment, and are able to work on source-code changes based on this information without necessarily having to engage in communication.

The team members in Ship used different forms of media to transmit information to each other. Instant messenger and email were both used to communicate, but for different purposes. Email was used to inquire about feature rationale and technical discussions, whereas meetings were used mostly to maintain task awareness of other team members.

The Team Mental Model of Open Communication Exchange

The Ship study revealed that the Ship team communicates openly, and that this benefited awareness and coordination. Team leaders communicated frequently with each other, even incorporating the environment coordinators in their discussions. I observed that experienced team members often kept aware of the events in the project to the point where they could help other team members when awareness breakdowns occurred. Although distance affected awareness, the shared task context and familiarity helped mitigate these effects.

Investigating their email patterns, particularly with respect to emergent people, revealed also that team members were concerned with exchanging information with each other efficiently. The methods in which they replied to each other and CCed messages suggested that they were interested in making each message helpful.

Thus, even though certain team members were new to the project, and even though the organization around the Ship team was undergoing an organizational restructuring, the team members knew who their team mates were and were very happy with the way that they were communicating.

Involvement of Important Communicators

In the Ship study, managers and support staff members were found to coordinate frequently, and were often in important positions within a social network, even though they rarely had assigned technical relationships in the network.

Developers and testers communicated frequently with each other, with most email traffic occurring between the team leaders and senior developers. However, the environment coordinators communicated with others multiple times and occupied relatively central locations in the social network; they were in a more central position than the managers were. Managers were found to send many more messages than they received. This suggests that these individuals are relevant to technical work,

and that conceptualizations of socio-technical congruence technical entities should be broadened to include these individuals.

Involvement of Emergent People

Emergent communicators are people who are involved in a message thread after the first message has been sent. They are important because they represent unknown participants in communication and changes in the relationships among project team members. I investigated emergent people in email discussions from the perspective of five technical team members. Emergent communicators were involved in a thread because it was an unexpected event, a request, an announcement, or a follow up. Emergent repliers replied because they were consulting with experts, providing status updates, reporting results, or providing expertise.

Emergent communicators are a special-case situation within a socio-technical environment. Their appearance does not always necessarily indicate a problem, but understanding the contexts that require bringing in emergent people may help practitioners and researchers understand which situations are a part of everyday life, and which situations are problems that lead to inefficiencies or to confusion.

The Context-Sensitive Effect of Socio-technical Congruence

In the RTC project, I studied the relationship between socio-technical congruence and team performance, conceptualized by the probability that a build succeeded.

I found an interaction effect between congruence and build result probability. If the build type was a continuous build, then increasing the congruence led to an increase in the build success probability. If the build type was an integration build, then increasing congruence actually led to a decrease in the build success probability. I also observed interaction effects between congruence and build date, the number of work items and build date, and build date and build type.

This finding illustrates that high congruence is not always beneficial—the conceptualizations are extremely important, and the context must be considered. In the case with integration builds, key points of contact could be included in the technical relationships network rather than everyone who contributed to the build.

A continuous build gathered file contributions and builds them into a single component, whereas an integration build involved components from multiple teams, built into the entire product. The teams that work one continuous builds therefore were more familiar with each other and were more able to communicate, as the builds do not require the involvement of as many people. However, in the case of the integration build, involving many people in the build may have actually harmed the build success. in RTC, they used a build coordinator whose purpose was explicitly to moderate discussion between different groups.

The choice of artifact appeared to have an effect on the influence of socio-technical congruence on an outcome. In this particular case, a coarse-grained artifact, such as an integration build, did not appear to reflect the desired property that high

congruence leads to a higher probability of build success. However, a fine-grained artifact, such as a continuous build, did reflect this property. The implication is that the choice of artifacts, and subsequently, the relationship of the nature of the dependency to people, determined whether socio-technical coordination is required to address technical issues.

The RTC study has broader importance than the relationship between socio-technical congruence and build success probability in RTC. The findings are important because they provide empirical evidence that illustrate the applicability of socio-technical congruence, and illustrates that an increase in socio-technical congruence may not always be connected to improvements in software development, depending on the conceptualizations and project context. Nonetheless, I believe that a socio-technical congruence model provides a useful method for conceptualizing and studying coordination, especially in a global team, and can be used to better understand the relationship between socio-technical congruence and performance in software development.

Communicating in Problem Situations

During the RTC case study, I studied gap size between team members. A large gap size indicated that team members were not bridging their technical relationships with communication. However, opposite of what I expected, the average gap sizes in successful builds was large, but that the average gap size in failed builds was small.

This suggested that developers were doing more communication in failed builds to cover technical dependencies. Developers appear to know their work, and collaborated more intensely when problems are prone to occur. As discussed in the RTC study and in related work, not all technical relationships needed to be bridged for the team members to complete their work; awareness through the environment was one way for them to coordinate. Thus, this suggests that there are work-related clues that provide developers with knowledge that certain tasks require more communication than others.

8.1.2 A Socio-technical Coordination Model

The goal of this dissertation was to create a socio-technical congruence model that presents refinements to conceptualizations of relationships, improves techniques with which to calculate alignment, and describes an approach that describes how to apply socio-technical congruence.

The model introduces three elements of socio-technical congruence that I describe improvements for: conceptualizations, techniques, and situation modelling. It incorporates a modified socio-technical congruence formula to model multiple relationships, and lays out a template that can be used for researchers to frame existing studies and to develop new studies. It also provides guidelines and considerations of how to conceptualize social relationships as well as technical relationships and deal with interesting and exceptional circumstances.

There are also three technical contributions in this model.

1. I proposed in this dissertation a weighted congruence measure, which assigns the relationships between people and artifacts, as well as artifacts and other artifacts, a numeric value. This allows someone applying the congruence framework to calculate the "gap size" between one person and another that identifies if the communication between those two people covers their technical relationships.

2. I presented the concept of emergent people in congruence. Team members are very perceptive of when certain individuals may not be receiving information in a timely fashion and are quick to remedy these situations.

   Involvement of someone outside of the team may require intervention if the emergent person is outside of the team but also outside of the task-oriented network of the team members. Any person who is not known to the team but also outside of the scope of their tasks may be an external contractor stakeholder making requests or a technical expert that was not included in the team. A manager may need to evaluate if this person's involvement is necessary or appropriate, and take measures to enable or disable this person's influence on the team.

3. I presented a technique to calculate multi-variable socio-technical congruence using different matrices. This allows certain types of relationships to "fill in" for other relationships. It also separates the variables into separate matrices, making management and calculation of socio-technical relationships easier.

The model brings to light the importance of choosing appropriate conceptualizations within a socio-technical congruence study. As socio-technical congruence is supposed to measure socio-technical coordination, the choice of conceptualizations and the understanding of a study's context affects one's conclusions greatly.

8.2 Applications to Research and Practice

The socio-technical congruence model has benefits to both research and to practice.

8.2.1 Applications to Software-Engineering Research

For researchers, this model describes circumstances that may occur in a socio-technical environment, and provides guidelines on how to handle these circumstances. A researcher who is not intimately familiar with details on an organization can use the model to represent a software team and identify potential factors that might affect software engineering outcomes, such as performance or quality. The techniques allow a researcher to modularly store and retrieve relationships between technical artifacts and people, and to combine them in a way that best reflect the socio-technical organization.

This model benefits a practitioner by allowing a software manager to identify what technical relationships exist within the organization, and to identify if an appropriate social relationship addresses this need. Gap calculations allow a practitioner to identify if a coordination need is not bridged by a relationship, whether it be through communication, or through a less-visible interaction like automatic awareness notifications or points of contact. A quantitative examination of relationships between team members may help confirm suspicions or concerns raised by the team. Depending on the nature of the problem, the remedy may be as simple as a reminder to a colleague to speak more often to a remote team member, or it may be as complicated as an organizational restructuring. In addition, the software industry stands to benefit from the increased understanding of coordination that can be achieved with this model.

8.2.2 Applications to Software-Engineering Practice

While this work is mainly targeted toward software-engineering researchers who intend to understand socio-technical communication, the findings and the framework developed in this model have implications on software practice, as well.

The socio-technical coordination model can be used by managers and consultants to better understand the coordination patterns of team members within a global software team. Such patterns can be automatically collected from a development environment. Multiple variables may be recorded and stored as separate relationships, and can be combined in multiple ways using the extensible socio-technical congruence analysis presented in Section 7.4.5. This allows the study of different relationships and their effect on coordination, with the intention of improving the efficiency of communication within the organization.

The finding that high congruence positively affected the probability of continuous builds succeeding, but that high congruence negatively affected the probability of integration builds suggests that socio-technical congruence has a limit of applicability as well, and that "perfect" congruence is not required for successful coordination. Thus, at this time, acquiring high congruence, under some conceptualizations, does not appear to be a necessary condition for success.

In addition, my observations of global software development has led to this short list of recommendations that can be used to benefit coordination. These observations include:

- A central issue-tracking repository reduces the need for synchronous communication and improves team member awareness of others' activities.
- Ensure that an experienced team member is familiar with the information needs of team members at other distributed sites.
- Hold regular meetings to ensure that team members are aware of the activities of others.

- When referring someone to another person in email, CC the new person as a courtesy to the discussants.
- Assign team members with multiple responsibilities so that their work cannot be blocked by delays in multi-site development.
- Be aware of emergent people and how they are related to your team members.

By simply being aware of emergent people, a manager or another important communicator may take a moment to identify who may be missing before sending important correspondence. Even something as simple as maintaining up-to-date mailing lists can reduce the delay introduced by emergent people in communication. The use of experienced team members can also help bridge communication between different teams.

8.3 Limitations

There are a number of limitations of this work. The first is that we base a number of the empirical findings on two projects only. While techniques were taken to increase internal validity, the external validity may be challenged. The findings from the individual studies are not generalisable at this time, but the findings of the awareness study were consistent with findings from previous literature. The findings from the communication patterns study and the socio-technical congruence study raised a number of interesting questions that should be considered in other cases.

Another limitation of this study is that the data sources, particularly for Ship, can not be made publicly available due to the sensitive nature of the information. In addition, the teams that I interacted with no longer exist because the company restructured the team, shifting the responsibilities of the individuals involved to other projects. As a result it was difficult for me to verify the results with the team as a whole, and it is not possible for this research to be replicated by other researchers.

Currently, the techniques that are used in the socio-technical congruence model rely on a number of external tools, including a tool that can extract work item relationships and change set relationships from a repository. Repository data extraction may be a technical hurdle to applying these techniques, and pose a significant barrier to allowing real-time extraction of socio-technical congruence data.

8.4 Future Work

This work leads to a number of future questions for investigation, both with respect to the empirical findings in the dissertation, and with respect to the validation of the proposed improvements to the socio-technical congruence model. As discussed in the limitations, a number of phenomenon were observed in the empirical investigations in this dissertation but must be generalized to a larger number of cases. However, the phenomenon observed in this dissertation can also be expanded into future work.

8.4.1 Brokers of Socio-technical Coordination

In this dissertation, I investigated emergent people, but one key component of the emergent person pattern that I did not investigate is the broker involved. Brokers or gatekeepers [108] are the intermediaries between groups and have been identified as holding key positions in a communication network [38, 209]. To further understand emergent people, one way would be to investigate the people who involve who these emergent people are in the first place.

Such a study would reveal what role within the software team a broker plays, as well as clarify if brokerage patterns occur frequently in email communication. This may lead to awareness on who team brokers are, how to identify team members with high potential to be brokers, and how to position these brokers to be effective at coordinating not only within the team, but outside of it. One outstanding question regards the work that such a broker does and whether it is congruent with communication—does a broker's technical work cross team boundaries like its communication does?

8.4.2 Validating the Emergent People Model

This preliminary analysis of emergent people in software development suggests some initial conceptualizations of their significance to the project based on *task* and *team* boundaries (Section 7.3.5). Emergent communicators represent an "unknown" element of the project. Future work needs to empirically validate my conceptualization.

One question is, are these emergent people participating as stakeholders in the project, or simply expert contributors? Does the involvement of emergent people that are previously unknown have a cost with respect to team performance? Whereas stakeholder identification is an area of current research [144], the prevalence of open repositories such as in RTC or in mailing lists mean that involvement from unknown individuals can be frequent. The goal of this future work would be to investigate an emergent person's significance in a project, and whether an emergent person must be included or acknowledged in project interactions. Classes of emergent people may be derived depending on how they are involved. These findings can affect requirements elicitation and analysis.

Another question is to examine the benefits of including emergent people. Is it cost-effective to reduce the number of emergent people in communication?

While our discussion of emergent people is in the context of a corporate environment where email is generally kept private, examining emergent people in more open projects may be valuable to learning about why particular individuals contribute expertise. In a team where information may be posted "open" to the organization, as in Rational Team Concert's work item database (Chapter 6), in mailing lists, or even to a public question-and-answer repository like Stack Overflow [207], examining emergent people may lead to the design of better knowledge-management systems where expert users are able to locate and answer questions for others.

8.4.3 Understanding Information Needs of Support Roles in Systems Organizations

The study of socio-technical coordination focused on software-development teams in industry. However, many of the findings may have applicability to other organization domains, including systems development.

More study of managerial responsibilities and their relationship to the features that must be developed on a technical level is beneficial to enabling these managers to work better. An empirically-derived understanding of a manager's relationship to the tasks on a project and a manager's need to communicate with team members is needed to effectively conceptualize a manager's relationship to technical entities.

Another role that is heavily involved in software development teams, but is often not investigated in research studies are the role of environment coordinators, information technology support staff, and other similar roles. In our studies, we were able to observe environment coordinators, but a more in-depth analysis would be able to identify their problems and tailor research customized to helping IT staff.

8.4.4 The Effect of Gaps in Socio-technical Coordination

The RTC empirical investigation is a first look at a weighted gap size measurement. A few previous studies have examined gaps (ex. [84, 148]) but more studies about their effects on a team is important. Marczak [148] suggests that not all gaps are detrimental to the project—that certain organizational constructs cover these gaps. Whether this is because gaps are not important, or if this is because of a poor conceptualization of technical relationships is not conclusive.

In addition, additional investigation into communication patterns of teams in crisis is needed. Two results in this dissertation suggest that crisis affects communication widely: The Ship empirical investigation reveals that large numbers of emergent people appear during crisis situations, and the RTC empirical investigation identified that gap sizes are smallest when builds are not successful. These results require further research.

8.4.5 Examining the Relationship Between Socio-technical Congruence and Task Complexity

The evidence in this dissertation suggests that there is a relationship between socio-technical congruence and build success in certain contexts. However, there are also instances where certain tasks cannot be completed even if congruence is high. This suggests that there may be characteristics of the task, or the quality of communication that affect the success criteria. For instance, one hypothesis for the results of the RTC study of gap sizes is that because the build was particularly complicated, more coordination among team members was needed, bridging the gap.

Future work in the area of the relationship between the fine-grained nature of the task as well as the quality of coordination would benefit our understanding of why socio-technical congruence is beneficial in certain situations, but not in others. A specific methodology would be to investigate builds that failed, but yet had high congruence in socio-technical congruence.

8.4.6 Evaluating the Multi-variable Socio-technical Coordination Model

While the model is meant to be generalisable in that it can incorporate most socio-technical coordination issues when investigating a software project, the model has not yet been applied to multiple software projects. In particular, the usefulness of the multi-variable aspect of the model has not been formally evaluated.

The multi-variable congruence model allows a researcher to combine multiple matrices for analysis. While this model was inspired as a result of empirical investigation of our two empirical investigations, and shows promise, it must be validated on further cases to prove its usefulness and applicability.

8.5 Conclusion

In this dissertation, I discussed the importance of socio-technical coordination when studying global software development teams. The goal of this research was to (1) refine existing conceptualizations of social and technical relationships, (2) overcome a limitation regarding dichotomized gap sizes, and (3) provide a set of guidelines for applying socio-technical congruence. I developed a socio-technical congruence model containing three elements: conceptualization, techniques, and situation modelling based on extant work. I proposed a gap size technique and a template for socio-technical congruence studies. However, there were still limitations about how to represent the complex social relationships between software development team members. These limitations were (1) the difficulty of conceptualizing awareness and (2) the difficulty of including relevant team members in the model. “” “” To address these limitations, I conducted two empirical investigations in a global team using observations, field notes, and email messages. The awareness study of the Ship team revealed that coordination is not always limited to expert coordination, that experience and familiarity with team members fills gaps, and that multiple forms of media are used simultaneously for coordination. The email communication patterns study of Ship revealed that important communicators include managers and environment coordinators—people who have no technical dependencies to code and are therefore not represented in most socio-technical congruence studies. I also examined emergent people and formed an initial model of emergent people based on task and team boundaries.

I executed a third empirical investigation to study socio-technical congruence in

a large, global team. I investigated the relationship between socio-technical coordination and team performance and applied the gap size measurement. I found that congruence was beneficial for certain types of builds only, and that high congruence can actually reduce build success probability for integration builds.

I improved the model to suggest refinements of conceptualizations of technical and social relationships and proposed how to handle *important communicators* as well as *emergent people*. An important communicator is usually a senior development leader or test leader, but managers and supporting staff are also heavily involved in interactions with team members. The socio-technical congruence model, as a result of the modifications, contains three elements: conceptualizations, alignment techniques, and situation modelling. It describes an accompanying process as well as a template for future studies. The model also contains a number of guidelines that can be used when representing socio-technical environments for studying socio-technical congruence. In addition to identifying a number of interesting phenomenon for future study, this dissertation has also provided guidelines and a template that one can use to for the study of socio-technical coordination in global software teams.

Though the concrete contribution of this dissertation is described in the form of the socio-technical congruence model as well as the empirical studies, one of the values of the model is that it brings to light the benefits and the drawbacks of using various conceptualizations for socio-technical congruence. It sparks an important discussion about the inherent theories behind socio-technical congruence. Intuitively, every team strives for "congruence", but it is important for a researcher to be able to identify high-quality conceptualizations of reality for a team while keeping the a model simple enough to be applicable to multiple cases. Thus, the model not only examines congruence at the fine-grained level of tasks and dependencies, but also allows one to take a step back to think about the "big picture" of coordination in a software organization.

References

[1] Design: Design Package [online]. Available from: http://cran.r-project.org/web/packages/Design/index.html [cited 2010-08-29].

[2] The R Project for Statistical Computing [online]. Available from: http://www.r-project.org/.

[3] CVS - Open Source Version Control [online]. 2006. Available from: http://www.nongnu.org/cvs/ [cited 2011-03-16].

[4] Email Addiction Survey [online]. 2008. Available from: http://o.aolcdn.com/cdn.webmail.aol.com/survey/aol/en-us/index.htm [cited December 1, 2010].

[5] 14,500 Customers and Counting - JIRA [online]. 2011. Available from: http://www.atlassian.com/software/jira/customers.jsp [cited 2011-03-16].

[6] Apache subversion [online]. 2011. Available from: http://subversion.apache.org/ [cited 2011-03-16].

[7] Bug, Issue, and Project Tracking for Software Development - JIRA [online]. 2011. Available from: http://www.atlassian.com/software/jira/ [cited 2011-03-16].

[8] Git - Fast Version Control System [online]. 2011. Available from: http://git-scm.com/ [cited 2011-03-16].

[9] Home :: Bugzilla [online]. 2011. Available from: http://www.bugzilla.org/ [cited 2011-03-16].

[10] Installation List [online]. 2011. Available from: http://www.bugzilla.org/installation-list/ [cited 2011-03-16].

[11] Mercurial SCM [online]. 2011. Available from: http://mercurial.selenic.com/ [cited 2011-03-16].

[12] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th international conference on World*

Wide Web, WWW '07, pages 835–844, New York, NY, USA, 2007. ACM. Available from: http://doi.acm.org/10.1145/1242572.1242685, doi:http://doi.acm.org/10.1145/1242572.1242685.

[13] Ban Al-Ani and H. Keith Edwards. A Comparative Empirical Study of Communication in Distributed and Collocated Development Teams. In *2008 IEEE International Conference on Global Software Engineering*, pages 35–44. Ieee, August 2008. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4638651, doi:10.1109/ICGSE.2008.9.

[14] Amer Al-Rawas and Steve Easterbrook. Communication problems in requirements engineering: A field study. In *Conference on Awareness in Software Engineering, London*, pages 47–60, February 1996.

[15] Thomas J. Allen. *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information Within the R&D Organization*. The MIT Press, January 1984. Available from: http://www.amazon.fr/exec/obidos/ASIN/0262510278/citeulike04-21.

[16] Chintan Amrit and Jos van Hillegersberg. Detecting coordination problems in collaborative software development environments. *Information Systems Management*, 25(1):57–70, December 2008.

[17] Jorge Aranda. *A Theory of Shared Understanding For Software Organizations*. PhD thesis, University of Toronto, 2010.

[18] Jorge Aranda and Gina Venolia. The secret life of bugs: Going past the errors and omissions in software repositories. *Software Engineering, International Conference on*, 0:298–308, 2009. doi:http://doi.ieeecomputersociety.org/10.1109/ICSE.2009.5070530.

[19] Gabriela Avram. Of deadlocks and peopleware - collaborative work practices in global software development. In *Proceedings of the International Conference on Global Software Engineering*, ICGSE '07, pages 91–102, Washington, DC, USA, 2007. IEEE Computer Society. Available from: http://dx.doi.org/10.1109/ICGSE.2007.30, doi:http://dx.doi.org/10.1109/ICGSE.2007.30.

[20] Gabriela Avram, Liam Bannon, John Bowers, Anne Sheehan, and Daniel K. Sullivan. Bridging, Patching and Keeping the Work Flowing: Defect Resolution in Distributed Software Development. *Computer Supported Cooperative Work (CSCW)*, 18(5-6):477–507, September 2009. Available from: http://www.springerlink.com/index/10.1007/s10606-009-9099-6, doi:10.1007/s10606-009-9099-6.

[21] Petra Badke-Schaub, Andre Neumann, Kristina Lauche, and Susan Mohammed. Mental models in design teams: a valid approach to performance in design collaboration. *CoDesign*, 3(1):5–20, 2007.

[22] Olle Bälter. Give the boss a break from email: Managers and their communication. In *In J Gulliksen, A Lantz, L Oestericher, K Severinson-Eklundh, Proceedings of NordiCHI 2000, STIMDI*, 2000.

[23] Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, number 5, pages 345–352. ACM, 2003. Available from: `http://portal.acm.org/citation.cfm?id=642672`.

[24] Georgine Beranek, Wolfgang Zuser, and Thomas Grechenig. Functional group roles in software engineering teams. *SIGSOFT Softw. Eng. Notes*, 30:1–7, May 2005. Available from: `http://doi.acm.org/10.1145/1082983.1083108`, `doi:http://doi.acm.org/10.1145/1082983.1083108`.

[25] Daniel M. Berry and Erik Kamsties. The syntactically dangerous all and plural in specifications. *IEEE Software*, 22:55–57, 2005. `doi:http://doi.ieeecomputersociety.org/10.1109/MS.2005.22`.

[26] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. In *Intl Conf on Software Engineering, Vancouver, Canada*, pages 518–528, May 2009.

[27] Christian Bird, Alex Gourley, and Anand Swaminathan. Mining email social networks in postgres. In *Mining Software Repositories Workshop 2006, ICSE*, 2006.

[28] Christian Bird, Nachiappan Nagappan, Premkumar Devanbu, Harald Gall, and Brendan Murphy. Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. *Communications of the ACM*, 52(8):85–93, August 2009.

[29] Christian Bird, Nachiappan Nagappan, Harald Gall, Brendan Murphy, and Premkumar Devanbu. Putting it all together: Using socio-technical networks to predict failures. *Software Reliability Engineering, International Symposium on*, 0:109–119, 2009. `doi:http://doi.ieeecomputersociety.org/10.1109/ISSRE.2009.17`.

[30] Christian Bird, David Pattison, Raissa D'Souza, Vladimir Filkov, and Premkumar Devanbu. Latent social structure in open source projects. In *SIGSOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 24–35, New York, NY, USA, 2008. ACM. `doi:http://doi.acm.org/10.1145/1453101.1453107`.

[31] Barry Boehm. A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 12–29, New York, NY, USA, 2006. ACM. Available from: http://doi.acm.org/10.1145/1134285.1134288, doi:http://doi.acm.org/10.1145/1134285.1134288.

[32] BW Boehm. Software engineering. *IEEE Transactions on Computers*, 25(12):1226–1241, 1976. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5010193.

[33] Jeremey Boissevain. *Friends of friends: Networks, manipulators and coalitions*. Wiley-Blackwell, 1974.

[34] Francesco Bolici, James Howison, and Kevin Crowston. Coordination without discussion? Socio-technical congruence and Stigmergy in Free and Open Source Software projects. In *Socio-Technical Congruence Workshop in conj ICSE 2009, Vancouver, Canada*, May 2009.

[35] S Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, January 2005. Available from: http://linkinghub.elsevier.com/retrieve/pii/S0378873304000693, doi:10.1016/j.socnet.2004.11.008.

[36] Robert P. Bostrom and J. Stephen Heinen. Mis problems and failures: A socio-technical perspective, part i: The causes. *MIS Quarterly*, 1(3):11–28, December 1977.

[37] David G. Boyer, Mark J. Handel, and James Herbsleb. Virtual community prescence awareness. *SIGGROUP Bull.*, 19:11–14, December 1998. Available from: http://doi.acm.org/10.1145/307736.307756, doi:http://doi.acm.org/10.1145/307736.307756.

[38] Ronald S. Burt. Structural holes and good ideas. *American Journal of Sociology*, 110(2):349–399, September 2004.

[39] Janis A. Cannon-Bowers, Eduardo Salas, and Sharolyn Converse. Shared mental models in expert team decision making. In Jr. N. John Castellan, editor, *Individual and Group Decision Making*, chapter 12, pages 221–246. Lawrence Erlbaum Associates, 1993.

[40] E. Carmel and R. Agarwal. Tactical approaches for alleviating distance in global software development. *Software, IEEE*, 18(2):22 –29, mar/apr 2001. doi:10.1109/52.914734.

[41] Erran Carmel. *Global Software Teams: Collaborating Across Borders and Time Zones*. Prentice Hall, 1999.

[42] S. Carter, J. Mankoff, and P. Goddi. Building connections among loosely coupled groups: Hebb's rule at work. *Computer Supported Cooperative Work*, 13(3-4):305–327, 2004. doi:http://dx.doi.org/10.1007/s10606-004-2805-5.

[43] M. Cataldo and S. Nambiar. Quality in Global Software Development Projects: A Closer Look at the Role of Distribution. In *Intl Conf on Global Software Engineering, Limerick, Ireland*, pages 163–172, July 2009. doi:10.1109/ICGSE.2009.24.

[44] Marcelo Cataldo. *Dependencies in geographically distributed software development: overcoming the limits of modularity.* PhD thesis, Carnegie Mellon University, 2007.

[45] Marcelo Cataldo, Matthew Bass, James D. Herbsleb, and Len Bass. On coordination mechanisms in global software development. In *ICGSE '07: Proceedings of the International Conference on Global Software Engineering*, pages 71–80, Washington, DC, USA, 2007. IEEE Computer Society. doi:http://dx.doi.org/10.1109/ICGSE.2007.33.

[46] Marcelo Cataldo, Matthew Bass, James D. Herbsleb, and Len Bass. On coordination mechanisms in global software development. In *Proceedings of the International Conference on Global Software Engineering*, ICGSE '07, pages 71–80, Washington, DC, USA, 2007. IEEE Computer Society. Available from: http://dx.doi.org/10.1109/ICGSE.2007.33, doi:http://dx.doi.org/10.1109/ICGSE.2007.33.

[47] Marcelo Cataldo and James D. Herbsleb. Communication networks in geographically distributed software development. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 579–588, New York, NY, USA, 2008. ACM. doi:http://doi.acm.org/10.1145/1460563.1460654.

[48] Marcelo Cataldo, James D. Herbsleb, and Kathleen M. Carley. Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity. In *Empirical Software Engineering Measurement, Kaiserslautern, Germany*, 2008.

[49] Marcelo Cataldo, Audris Mockus, Jeffrey A. Roberts, and James D. Herbsleb. Software dependencies, work dependencies, and their impact on failures. *IEEE Transactions on Software Engineering*, 42, 2009. doi:http://doi.ieeecomputersociety.org/10.1109/TSE.2009.42.

[50] Marcelo Cataldo, Patrick A. Wagstrom, James D. Herbsleb, and Kathleen M. Carley. Identification of coordination requirements: Implications for the design of collaboration and awareness tools. In *Conf on Computer-supported Cooperative Work, Banff, Canada*, October 2006.

[51] A. Cerveny Daniel and P. Robert. A study of the effects of three commonly used software engineering strategies on software enhancement productivity. *Information & Management*, 14(5):243–251, 1988. Available from: http://linkinghub.elsevier.com/retrieve/pii/0378720688900122.

[52] Anurat Chapanond, Mukkai S. Krishnamoorthy, and Bülent Yener. Graph theoretic and spectral analysis of enron email data. *Computational and Mathematical Organization Theory*, 11(3):265–281, January 2005. Available from: http://www.springerlink.com/index/10.1007/s10588-005-5381-4, doi:10.1007/s10588-005-5381-4.

[53] Lee Li-Jen Chen and Brian R. Gaines. A CyberOrganism model for awareness in collaborative communities on the Internet. *International Journal of Intelligent Systems*, 12(1):31–56, January 1997. Available from: http://doi.wiley.com/10.1002/(SICI)1098-111X(199701)12:1<31::AID-INT2>3.0.CO;2-Z, doi:10.1002/(SICI)1098-111X(199701)12:1<31::AID-INT2>3.0.CO;2-Z.

[54] Li-Te Cheng, Susanne Hupfer, Steven Ross, and John Patterson. Jazzing up eclipse with collaborative tools. In *Eclipse '03: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, pages 45–49, New York, NY, USA, 2003. ACM Press. doi:http://doi.acm.org/10.1145/965660.965670.

[55] Albert Cherns. The Principles of Sociotechnical Design. *Human Relations*, 29(8):783–792, 1976. Available from: http://hum.sagepub.com/content/29/8/783.short, arXiv:http://hum.sagepub.com/content/29/8/783.full.pdf+html, doi:10.1177/001872677602900806.

[56] M. Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[57] Nancy J. Cooke, Eduardo Salas, Janis A. Cannon-Bowers, and Rene'e Stout. Measuring team knowledge. *Human Factors*, 42(1):151–173, 2000.

[58] Peter Cooper. whatlanguage [online]. Available from: http://github.com/peterc/whatlanguage/tree/master [cited December 2, 2010].

[59] Catherine D. Cramton. The mutual knowledge problem and its consequences for dispersed collaboration. *Organization Science*, 12(3):346–371, 2001.

[60] G. Creamer, Ryan Rowe, Shlomo Hershkop, and S. Stolfo. Segmentation and automated social hierarchy detection through email network analysis. *Advances in Web Mining and Web Usage Analysis*, pages 40–58, 2009. Available from: http://www.springerlink.com/index/a861t12082l1l67.pdf.

[61] John W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications, 2nd edition, 2002.

[62] Kevin Crowston and James Howison. The social structure of free and open source software development. *First Monday*, 10(2), February 2005. http://firstmonday.org/issues/issue10_2/crowston/index.html.

[63] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. Available from: `http://igraph.sf.net`.

[64] Bill Curtis, Herb Krasner, and Neil Iscoe. A field study of the software design process for large systems. *Commun. ACM*, 31(11):1268–1287, 1988. `doi:http://doi.acm.org/10.1145/50087.50089`.

[65] L.A. Dabbish and R.E. Kraut. Email overload at work: an analysis of factors associated with email strain. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 431–440. ACM, 2006. Available from: `http://portal.acm.org/citation.cfm?id=1180875.1180941`.

[66] Laura a. Dabbish, Robert E. Kraut, Susan Fussell, and Sara Kiesler. Understanding email use: predicting action on a message. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '05*, page 691, New York, New York, USA, 2005. ACM Press. Available from: `http://portal.acm.org/citation.cfm?doid=1054972.1055068`, `doi:10.1145/1054972.1055068`.

[67] Barthélémy Dagenais, Harold Ossher, R.K.E. Bellamy, M.P. Robillard, and J.P. de Vries. Moving into a new software project landscape. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 275–284. ACM, 2010. Available from: `http://portal.acm.org/citation.cfm?id=1806799.1806842`.

[68] Åsa G. Dahlstedt and Anne Persson. *Requirements Interdependencies: State of the Art and Future Challenges*, chapter 5, pages 95–116. Springer Berlin Heidelberg, 2006.

[69] Daniela Damian, Luis Izquierdo, Janice Singer, and Irwin Kwan. Awareness in the wild: Why communication breakdowns occur. In *Intl Conf on Global Software Engineering, Münich, Germany*, pages 81–90, August 2007.

[70] Daniela Damian, Irwin Kwan, and Sabrina Marczak. Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people. In Jim Whitehead, Ivan Mistrík, John Grundy, and Andr'e van der Hoek, editors, *Collaborative Software Engineering*, chapter 3, pages 57–76. Springer-Verlag, 2010.

[71] Daniela Damian, Sabrina Marczak, Madalina Dascalu, Michael Heiss, and Adrian Liche. Using a real-time conferencing tool in distributed collaboration: An experience report from siemens it solutions and services. In *Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*, ICGSE '09, pages 239–243, Washington, DC, USA, 2009. IEEE Computer Society. Available from: `http://dx.doi.org/10.1109/ICGSE.2009.31`, `doi:http://dx.doi.org/10.1109/ICGSE.2009.31`.

[72] Daniela Damian, Sabrina Marczak, and Irwin Kwan. Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centered Social Networks. In *Intl Requirements Engineering Conference, New Delhi, India*, October 2007.

[73] Daniela E. Damian and Didar Zowghi. The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In *IEEE Joint International Conference on Requirements Engineering 2002, September 9–13*, pages 319–328, September 2002.

[74] Catalina Danis, W.A. Kellogg, Tessa Lau, Mark Dredze, Jeffrey Stylos, and Nicholas Kushmerick. Managers' email: beyond tasks and to-dos. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1324–1327. ACM, 2005. Available from: `http://portal.acm.org/citation.cfm?id=1056808.1056907`.

[75] Cleidson de Souza, Paul Dourish, David Redmiles, Stephen Quirk, and Erik Trainer. From technical dependencies to social dependencies. In *CSCW'04: Workshop on Social Networks*, 2004.

[76] Cleidson R. B. de Souza and David Redmiles. The Awareness Network: To Whom Should I Display My Actions? And, Whose Actions Should I Monitor? In *European Conf on Computer Supported Cooperative Work, Limerick, Ireland*, September 2007.

[77] Cleidson R. B. de Souza and David F. Redmiles. An empirical study of software developers' management of dependecies and changes. In *International Conference on Software Engineering, Leipzig, Germany*, May 2008.

[78] Cleidson R. B. d. de Souza, David Redmiles, Li-Te Cheng, David Millen, and John Patterson. How a good software practice thwarts collaboration: the multiple roles of apis in software development. *Software Engineering Notes*, pages 221–230, 2004. `doi:10.1145/1029894.1029925`.

[79] Jana Diesner, Terrill L. Frantz, and Kathleen M. Carley. Communication Networks from the Enron Email Corpus It's Always About the People. Enron is no Different. *Computational and Mathematical Organization Theory*, 11(3):201–228, January 2006. Available from: `http://www.springerlink.com/index/10.1007/s10588-005-5377-0`, `doi:10.1007/s10588-005-5377-0`.

[80] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 107–114, New York, NY, USA, 1992. ACM Press. doi:http://doi.acm.org/10.1145/143457.143468.

[81] Paul Dourish and Sara Bly. Portholes: supporting awareness in a distributed work group. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 541–547, New York, NY, USA, 1992. ACM. Available from: http://doi.acm.org/10.1145/142750.142982, doi:http://doi.acm.org/10.1145/142750.142982.

[82] Yael Dubinsky and Orit Hazzan. Roles in agile software development teams. In Jutta Eckstein and Hubert Baumeister, editors, *Extreme Programming and Agile Processes in Software Engineering*, volume 3092 of *Lecture Notes in Computer Science*, pages 157–165. Springer Berlin / Heidelberg, 2004.

[83] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer London, 2008.

[84] K. Ehrlich, M. Helander, G. Valetto, S. Davies, and C. Williams. An analysis of congruence gaps and their effect on distributed software development. In *Socio-Technical Congruence Workshop in conj. ICSE 2008, Leipzig, Germany*, 2008.

[85] Kate Ehrlich and Klarissa Chang. Leveraging expertise in global software teams: Going outside boundaries. In *Intl Conf on Global Software Engineering 2006, Florianopolis, Brazil*, pages 149–158, October 2006.

[86] T. Eisenbarth, R. Koschke, and D. Simon. Aiding program comprehension by static and dynamic feature analysis. In *Software Maintenance, 2001. Proceedings. IEEE International Conference on*, pages 602 –611, 2001. doi:10.1109/ICSM.2001.972777.

[87] Elliot E. Entin and Daniel Serfaty. Adaptive Team Coordination. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 41(2):312–325, June 1999. Available from: http://hfs.sagepub.com/cgi/doi/10.1518/001872099779591196, doi:10.1518/001872099779591196.

[88] Alberto Espinosa, Robert Kraut, Sandra Slaughter, Javier Lerch, and James Herbsleb. Shared mental models, familiarity, and coordination: A multi-method study of distributed software teams. In *International Conference on Information Systems (ICIS)*, number 39, 2002.

[89] J. Alberto Espinosa, Sandra A. Slaughter, Robert E. Kraut, and James D. Herbsleb. Team knowledge and coordination in geographically distributed software development. *Journal of Management Information Systems*, 24(1):135–169, 2007.

[90] Jr. F. P. Brooks. *The Mythical Man-Month.* Addison-Wesley, 1975.

[91] Samer Faraj and Lee Sproull. Coordinating Expertise in Software Development Teams. *Management Science*, 46(12):1554–1568, 2000. doi:10.1287/mnsc.46.12.1554.12072.

[92] Danyel Fisher, AJ Brush, Eric Gleave, and M.A. Smith. Revisiting Whittaker & Sidner's "Email Overload" ten years later. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 309–312. ACM, 2006. Available from: http://portal.acm.org/citation.cfm?id=1180875.1180922.

[93] Michael Freed, Jaime Carbonell, Geoff Gordon, Jordan Hayes, Brad Myers, Daniel Siewiorek, Stephen Smith, Aaron Steinfeld, and Anthony Tomasic. Radar: A personal assistant that learns to reduce email overload. In *Proc. of AAAI Conference on Artificial Intelligence*, pages 15–21, 2008. Available from: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:RADAR+:+A+Personal+Assistant+that+Learns+to+Reduce+Email+Overload#0.

[94] Linton C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1(3):215–239, 1978/1979.

[95] Randall Frost. Jazz and the eclipse way of collaboration. *IEEE Software*, 24(06):114–117, 2007.

[96] Hugo Fuks, Alberto B. Raposo, Marco A. Gerosa, and Carlos J. P. Lucena. Applying the 3c model to groupware development. *International Journal of Cooperative Information Systems*, 2005.

[97] Susan R. Fussell, Robert E. Kraut, F. Javier Lerch, William L. Scherlis, Matthew M. McNally, and Jonathan J. Cadiz. Coordination, overload and team performance: effects of team communication strategies. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, CSCW '98, pages 275–284, New York, NY, USA, 1998. ACM. Available from: http://doi.acm.org/10.1145/289444.289502, doi:http://doi.acm.org/10.1145/289444.289502.

[98] H. Gall, K. Hajek, and M. Jazayeri. Detection of logical coupling based on product release history. In *Proceedings of the International Conference on Software Maintenance, Bethesda, USA.*, pages 190–198, 1998.

[99] Laura Garton and Barry Wellman. Social Impacts of Electronic Mail in Organizations: A Review of the Research Literature. *Communications Yearbook*, 18(19):435—-453, 1995.

[100] S. Geister. Effects of Process Feedback on Motivation, Satisfaction, and Performance in Virtual Teams. *Small Group Research*, 37(5):459–489, October 2006. Available from: http://sgr.sagepub.com/cgi/doi/10.1177/1046496406292337, doi:10.1177/1046496406292337.

[101] Bilal Gokpinar, Wallace J. Hopp, and Seyed M. R. Iravani. The Impact of Misalignment of Organizational Structure and Product Architecture on Quality in Complex Product Development. *Management Science*, 2010. Available from: http://mansci.journal.informs.org/cgi/content/abstract/56/3/468, doi:10.1287/mnsc.1090.1117.

[102] V. Gruhn and J. Urbainczyk. Software process modeling and enactment: an experience report related to problem tracking in an industrial project. In *Software Engineering, 1998. Proceedings of the 1998 International Conference on*, pages 13 –21, April 1998. doi:10.1109/ICSE.1998.671098.

[103] Carl Gutwin, Reagan Penner, and Kevin Schneider. Group awareness in distributed software development. In *Conf on Computer-supported Cooperative Work, Chicago, USA*, pages 72–81, November 2004. doi:http://doi.acm.org/10.1145/1031607.1031621.

[104] Carl Gutwin, Kevin A. Schneider, David Paquette, and Reagan Penner. *Supporting Group Awareness in Distributed Software Development.*, volume 3425 of *Lecture Notes in Computer Science*, pages 383–397. Springer, 2004. Available from: http://www.springerlink.com/content/32a2t0vcq15yw5qv/.

[105] Carl Gutwin, Gwen Stark, and Saul Greenberg. Support for workspace awareness in educational groupware. In *The first international conference on Computer support for collaborative learning*, CSCL '95, pages 147–156, Hillsdale, NJ, USA, 1995. L. Erlbaum Associates Inc. Available from: http://portal.acm.org/citation.cfm?id=222020.222126.

[106] Mark Handel and James D. Herbsleb. What is chat doing in the workplace? In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, CSCW '02, pages 1–10, New York, NY, USA, 2002. ACM. Available from: http://doi.acm.org/10.1145/587078.587080, doi:http://doi.acm.org/10.1145/587078.587080.

[107] Ahmed E. Hassan and Ken Zhang. Using decision trees to predict the certification result of a build. In *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, pages 189–198, Washington,

DC, USA, 2006. IEEE Computer Society. Available from: http://portal.acm.org/citation.cfm?id=1169218.1169318, doi:10.1109/ASE.2006.72.

[108] Jurgen Hauschildt and Gerhard Schewe. Gatekeeper and process promotor: key persons in agile and innovative organizations. *International Journal of Agile Management Systems*, 2(2), 2000. Available from: http://proquest.umi.com/pqdweb?did=115723426&Fmt=7&clientId=65345&RQT=309&VName=PQD.

[109] Christian Heath, Marcus S. Svensson, Jon Hindmarsh, Paul Luff, and Dirk V. Lehn. Configuring awareness. *Comput. Supported Coop. Work*, 11(3):317–347, 2002. Available from: http://portal.acm.org/citation.cfm?id=586346, doi:10.1023/A:1021247413718.

[110] Rebecca M. Henderson and Kim B. Clark. Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, 35:9–30, March 1990.

[111] J. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on*, 29(6):481–494, 2003.

[112] James D. Herbsleb. Global software engineering: The future of socio-technical coordination. In *Future of Software Engineering in conj. ICSE 2007, Minneapolis, USA*, pages 188–198, 2007. doi:http://dx.doi.org/10.1109/FOSE.2007.11.

[113] James D. Herbsleb and Rebecca E. Grinter. Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software*, 16(5):63–70, 1999. doi:http://dx.doi.org/10.1109/52.795103.

[114] James D. Herbsleb and Rebecca E. Grinter. Splitting the organization and integrating the code: Conway's law revisited. In *ICSE*, pages 85–95, 1999.

[115] James D. Herbsleb, Helen Klein, Gary M. Olson, Hans Brunner, Judish S. Olson, and Joe Harding. Object-oriented analysis and design in software project teams. *Human-Computer Interaction*, 10(2/3):249–293, 1995.

[116] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development: Distance and speed. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 81–90, 2001.

[117] Abram Hindle, Neil A. Ernst, Michael W. Godfrey, and John Mylopoulos. Automated topic naming to support cross-project analysis of software maintenance activities. In *Mining Software Repositories 2011, Honolulu, USA, May 21–22*, 2011.

[118] Pamela Hinds and Cathleen McGrath. Structures that Work: Social Structure, Work Structure and Coordination Ease in Geographically Distributed Teams. In *Conf on Computer Supported Cooperative Work*, pages 343–352, New York, NY, USA, 2006. ACM. doi:http://doi.acm.org/10.1145/1180875.1180928.

[119] Helena Holmstrom, Eoin Ó Conchúir, Pär J øAgerfalk, and Brian Fitzgerald. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In *ICGSE 2006: First IEEE International Conference on Global Software Engineering*, pages 3–11, 2006.

[120] Liaquat Hossain, Andre Wu, and Kenneth K S Chung. Actor centrality correlates to project based coordination. In *Computer-supported Cooperative Work, Banff, Canada*, 2006.

[121] James Howison, Keisuke Inoue, and Kevin Crowston. Social dynamics of free and open source team communications. In *Second Intl Conf on Open Source Systems*, Como, Italy, June 2006.

[122] Yvonne Hsieh. Culture and shared understanding in distributed requirements engineering. In *First International Conference on Global Software Engineering, Florianópolis, Brazil, October*, pages 101–105, 2006.

[123] Edwin Hutchins. *Cognition in the wild*. MIT Press, 1995.

[124] Edwin Hutchins and Tove Klausen. Distributed cognition in an airline cockpit. In Yrjö Engeström and David Middleton, editors, *Cognition and Communication At Work*, Cognition and Communication at Work, chapter 2, pages 15–34. Cambridge University Press, 1998.

[125] Ellen Isaacs, Alan Walendowski, Steve Whittaker, Diane J. Schiano, and Candace Kamm. The character, functions, and styles of instant messaging in the workplace. *Proceedings of the 2002 ACM conference on Computer supported cooperative work - CSCW '02*, page 11, 2002. Available from: http://portal.acm.org/citation.cfm?doid=587078.587081, doi:10.1145/587078.587081.

[126] Ellen Issacs, Candace Kamm, Diane J. Schiano, Alan Walendowski, and Steve Whittaker. Characterizing instant messaging from recorded logs. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, 2002.

[127] T.W. Jackson and Ray Dawson. Understanding email interaction increases organizational productivity. *Communications of the ACM*, 46(8):84, 2003. Available from: http://portal.acm.org/citation.cfm?id=859670.859673.

[128] Thomas Karagiannis and M. Vojnovic. Behavioral profiles for advanced email features. In *Proceedings of the 18th international conference on World wide web*, pages 711–720. ACM, 2009. Available from: http://portal.acm.org/citation.cfm?id=1526709.1526805.

[129] P. S. Keila. Structure in the Enron Email Dataset. *Computational and Mathematical Organization Theory*, 12(3):63–199, October 2005. doi:10.1007/s10588-005-5379-y.

[130] Mumtaz Muhammad Khan, Sulaiman Aziz Lodhi, and Mumannad Abdul Majid Makk. Measuring team implicit coordination. *Australian Journal of Basic and Applied Sciences*, 4(6):1211–1136, 2010.

[131] R. Klimoski and S. Mohammed. Team Mental Model: Construct or Metaphor? *Journal of Management*, 20(2):403–437, April 1994. Available from: http://jom.sagepub.com/cgi/doi/10.1177/014920639402000206, doi:10.1177/014920639402000206.

[132] Andrew J. Ko, Robert DeLine, and Gina Venolia. Information needs in collocated software development teams. In *Intl Conf on Software Engineering, Minneapolis, USA*, pages 344–353, 2007. doi:http://dx.doi.org/10.1109/ICSE.2007.45.

[133] R. Kraut and L. Streeter. Coordination in software development. *Commun. ACM*, 38(3):69–81, March 1995.

[134] Robert E. Kraut and Paul Attewell. *Media use in a global corporation: Electronic mail and organizational knowledge*, chapter 15, pages 323–342. Lawrence Erlbaum Associates, 1997. Email overload may be a myth. Email allows people to be more aware of what's going on in the corporation. Email is not as intrusive as other media.

[135] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 611–617, New York, NY, USA, 2006. ACM. Available from: http://doi.acm.org/10.1145/1150402.1150476, doi:http://doi.acm.org/10.1145/1150402.1150476.

[136] Irwin Kwan and Daniela Damian. Extending socio-technical congruence with awareness relationships. In *SSE 2011: Fourth International Workshop on Social Software Engineering in conj. joint meeting of ESFC/FSE, Szeged, Hungary*, 2011.

[137] Irwin Kwan and Daniela Damian. The hidden experts in software-engineering communication (nier track). In *ICSE'11: Proceedings of the Intl Conf on Software Engineering, Honolulu, USA*, 2011.

[138] Irwin Kwan, Daniela Damian, and Sabrina Marczak. The effects of distance, experience, and communication structure on requirements awareness in two distributed industrial software projects. In *Global Requirements Engineering Workshop (GREW'07), in Proc. of International Conference on Global Software Engineering (ICGSE 2007), Munich, Germany*, 2007.

[139] Irwin Kwan, Adrian Schröter, and Daniela Damian. A weighted congruence measure. In *Socio-Technical Congruence Workshop in conj. ICSE 2009, Vancouver, Canada*, 2009.

[140] Irwin Kwan, Adrian Schroter, and Daniela Damian. Does socio-technical congruence have an effect on software build success? a study of coordination in a software project. *IEEE Transactions on Software Engineering*, 37(3):307–324, 2011. doi:http://doi.ieeecomputersociety.org/10.1109/TSE.2011.29.

[141] Matthew J. LaMantia, Yuanfang Cai, Alan MacCormack, and John Rusnak. Analyzing the Evolution of Large-Scale Software Systems Using Design Structure Matrices and Design Rule Theory: Two Exploratory Cases. *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, pages 83–92, February 2008. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4459146, doi:10.1109/WICSA.2008.49.

[142] Marek Leszak and Manfred Meier. Successful global development of a large-scale embedded telecommunications product. In *ICGSE 2007: Second IEEE International Conference on Global Software Engineering*, pages 23–32, 2007.

[143] T.C. Lethbridge, Janice Singer, and Andrew Forward. How software engineers use documentation: the state of the practice. *Software, IEEE*, 20(6):35–39, 2003. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1241364.

[144] 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 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 295–304, New York, NY, USA, 2010. ACM. Available from: http://doi.acm.org/10.1145/1806799.1806844, doi:http://doi.acm.org/10.1145/1806799.1806844.

[145] Alan MacCormack, John Rusnak, and Carliss Y. Baldwin. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7):1015–1030, July 2006.

[146] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Comput. Surv.*, 26(1):87–119, March 1994. Available from: http://portal.acm.org/citation.cfm?id=174668, doi:10.1145/174666.174668.

[147] James March and Herbert Simon. *Organizations*. Blackwell Publishers, 238 Main Street, Suite 501, Cambridge, Massachusetts, USA, 2nd edition, 1958.

[148] Sabrina Marczak. *On the Understanding of Requirements-Driven Collaboration: A Framework and an Empirical Field Investigation*. PhD thesis, University of Victoria, 3800 Finnerty Road, Victoria, BC, February 2011.

[149] Sabrina Marczak, Daniela Damian, and Irwin Kwan. Assessing collaboration driven by requirements: A role-based. In submission, 2011.

[150] Sabrina Marczak, Daniela Damian, Ulrike Stege, and Adrian Schröter. Information Brokers in Requirement-Dependency Social Networks. *2008 16th IEEE International Requirements Engineering Conference*, pages 53–62, September 2008. Available from: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4685652`, `doi:10.1109/RE.2008.26`.

[151] Sabrina Marczak, Daniela Damian, Ulrike Stege, and Adrian Schröter. Information brokers in requirement-dependent social networks. In *Intl Conf on Requirements Engineering, Barcelona, Spain*, September 2008.

[152] Sabrina Marczak, Irwin Kwan, and Daniela Damian. Investigating collaboration driven by requirements investigating collaboration driven by requirements in cross-functional software teams. In *Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills (CIRCUS), in conj. RE 2009, Atlanta, USA.*, August 2009.

[153] John Mathieu, Travis Maynard, Tammy Rapp, and Lucy Gilson. Team effectiveness 1997-2007: A review of recent advancements and a glimpse into the future. *Journal of Management*, 34(3):410–476, June 2008. `http://jom.sagepub.com/content/34/3/410`.

[154] M.L. Maznevski and K.M. Chudoba. Bridging Space over time: Global Virtual Team Dynamics and Effectiveness. *Organization science*, 11(5):473–492, 2000. Available from: `http://www.jstor.org/stable/2640340`.

[155] David W. McDonald and Mark S. Ackerman. Expertise recommender: a flexible recommendation system and architecture. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 231–240, New York, NY, USA, 2000. ACM. Available from: `http://doi.acm.org/10.1145/358916.358994`, `doi:http://doi.acm.org/10.1145/358916.358994`.

[156] A.E. Milewski, M. Tremaine, R. Egan, S. Zhang, F. Kobler, and P. O'Sullivan. Guidelines for effective bridging in global software engineering. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*, pages 23 –32, 2008. `doi:10.1109/ICGSE.2008.16`.

[157] Shawn Minto and Gail C. Murphy. Recommending emergent teams. In *Mining Software Repositories, 2007. ICSE Workshops MSR '07. Fourth International Workshop on*, pages 5–5, 2007. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4228642.

[158] Audris Mockus, Roy T. Fielding, and James Herbsleb. A case study of open source software development: the apache server. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 263–272, New York, NY, USA, 2000. ACM Press. doi:http://doi.acm.org/10.1145/337180.337209.

[159] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *Software Eng. Methodology, ACM Trans. on*, 11(3):309–346, 2002. doi:http://doi.acm.org/10.1145/567793.567795.

[160] Audris Mockus and James D. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 503–512, New York, NY, USA, 2002. ACM. Available from: http://doi.acm.org/10.1145/581339.581401, doi:http://doi.acm.org/10.1145/581339.581401.

[161] S. Mohammed, L. Ferzandi, and K. Hamilton. Metaphor No More: A 15-Year Review of the Team Mental Model Construct. *Journal of Management*, 36(4):876–910, February 2010. Available from: http://jom.sagepub.com/cgi/doi/10.1177/0149206309356804, doi:10.1177/0149206309356804.

[162] Nachiappan Nagappan, Brendan Murphy, and Victor Basili. The Influence of Organizational Structure on Software Quality: An Empirical Case Study. In *Intl Conf on Software Engineering, Leipzig, Germany*, pages 521–530, May 2008. doi:http://doi.acm.org/10.1145/1368088.1368160.

[163] K. Nakakoji, Y. Ye, and Y. Yamamoto. *Supporting Expertise Communication in Developer-Centered Collaborative Software Development Environments*, chapter 11. Springer-Verlag, 2010.

[164] D.C. Neale, J.M. Carroll, and M.B. Rosson. Evaluating computer-supported cooperative work: models and frameworks. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 112–121. ACM, 2004. Available from: http://portal.acm.org/citation.cfm?id=1031626.

[165] M. E. J. Newman. Mathematics of networks. In L. E. Blume and S. N. Durlauf, editors, *The New Palgrave Encyclopedia of Economics*. Palgrave Macmillan, Basingstoke, 2008.

[166] Thanh Nguyen, Timo Wolf, and Daniela Damian. Global software development and delay: Does distance still matter? In *Intl Conf on Global Software Engineering, Bangalore, India*, pages 45–54, August 2008. doi:http://doi.ieeecomputersociety.org/10.1109/ICGSE.2008.39.

[167] Brian Nicholson and Sundeep Sahay. Some political and cultural issues in the globalisation of software development: case experience from britain and india. *Information and Organization*, 11(1):25 – 43, 2001. Available from: http://www.sciencedirect.com/science/article/B6W7M-422G6PW-3/2/9e9e0d29d8a4f3b0d3bafb7bb78d1648, doi:DOI:10.1016/S0959-8022(00)00008-4.

[168] Tuomas Niinimäki and Casper Lassenius. Experiences of instant messaging in global software development projects: A multiple case study. *Intl Conf on Global Software Engineering, Bangalore, India*, pages 55–64, July 2008. doi:http://doi.ieeecomputersociety.org/10.1109/ICGSE.2008.27.

[169] Kjell Ivar Ø vergård, Cato Alexander Bjø rkli, Bjarte Knappen Rø ed, and Thomas Hoff. Control strategies used by experienced marine navigators: observation of verbal conversations during navigation training. *Cognition, Technology & Work*, 12(3):163–179, August 2009. Available from: http://www.springerlink.com/index/10.1007/s10111-009-0132-9, doi:10.1007/s10111-009-0132-9.

[170] Briony J. Oates. *Researching Information Systems and Computing*. SAGE Publications, 2006.

[171] Rocco Oliveto, Malcom Gethers, Gabriele Bavota, Denys Poshyvanyk, and Andrea De Lucia. Identifying method friendships to remove the feature envy bad smell. In *International Conference on Software Engineering '11, Honolulu, USA*, 2011.

[172] R. Oser, C. Prince, B. B. Morgan Jr., and S. S. Simpson. An analysis of aircrew communication patterns and content. Technical Report 90-009, Naval Training Systems Center, 1991. Available from http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA246618Location=U2doc=GetTRDoc.pdf.

[173] Lucas Panjer, Irwin Kwan, and Daniela Damian. Jazz team coordination tools. In *Jazz Birds of a Feather (poster session) in conj. OOPSLA 2007, Montreal, Canada*, 2007.

[174] D. L. Parnas. On the criteria to be used in decomposing systems into modules. 15(12):1053–1058, December 1972. Available from: http://portal.acm.org/citation.cfm?id=1241515.1241527.

[175] Dewayne E. Perry, Nancy A. Staudenmayer, and Lawrence G. Votta. People, organizations, and process improvement. *IEEE Software*, 11(4):36–45, 1994.

[176] Shari Lawrence Pfleeger. *Software Engineering: Theory and Practice*. Prentice Hall, Upper Saddle River, New Jersey, second edition, 2001.

[177] J Phillips and L Reddie. Decisional style and self-reported Email use in the workplace. *Computers in Human Behavior*, 23(5):2414–2428, September 2007. Available from: http://linkinghub.elsevier.com/retrieve/pii/S0747563206000574, doi:10.1016/j.chb.2006.03.016.

[178] Christopher Poile. The Echo Method : Investigating socio-technical interactions. In *Socio-technical Congruence Workshop*, 2008.

[179] R. Prikladnicki, J.L.N. Audy, D. Damian, and T.C. de Oliveira. Distributed software development: Practices and challenges in different business strategies of offshoring and onshoring. In *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*, pages 262 –274, 2007. doi:10.1109/ICGSE.2007.19.

[180] Ronald E. Rice and JosephJack Torobin Grant, August E.Schmitz. Individual and network influences on the adoption and perceived outcomes of electronic messaging. *Social Networks*, 12(1):27–55, March 1990. Available from: http://linkinghub.elsevier.com/retrieve/pii/037887339090021Z, doi:10.1016/0378-8733(90)90021-Z.

[181] Peter C. Rigby and Margaret-Anne Storey. Understanding broadcast based peer review on open source software projects. In *Proceeding of the 33rd international conference on Software engineering*, ICSE '11, pages 541–550, New York, NY, USA, 2011. ACM. Available from: http://doi.acm.org/10.1145/1985793.1985867, doi:http://doi.acm.org/10.1145/1985793.1985867.

[182] W.B. Rouse, J.a. Cannon-Bowers, and E. Salas. The role of mental models in team performance in complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1296–1308, 1992. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=199457, doi:10.1109/21.199457.

[183] Ryan Rowe, G. Creamer, S. Hershkop, and S.J. Stolfo. Automated social hierarchy detection through email network analysis. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 109–117. ACM, 2007. Available from: http://portal.acm.org/citation.cfm?id=1348549.1348562.

[184] E. Salas, T. L. Dickinson, S. A. Converse, and S. I. Tannenbaum. Toward an understanding of team performance and training. In R. W. Swezey and E. Salas, editors, *Teams: Their training and performance*, pages 3–29. Ablex, 1992.

[185] Neeraj Sangal, Ev Jordan, Vineet Sinha, and Daniel Jackson. Using dependency models to manage complex software architecture. *SIGPLAN Not.*, 40:167–176, October 2005. Available from: http://doi.acm.org/10.1145/1103845.1094824, doi:http://doi.acm.org/10.1145/1103845.1094824.

[186] A. Sarma, Z. Noroozi, and A. van der Hoek. Palantir: raising awareness among configuration management workspaces. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 444–454, 2003. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1201222.

[187] Anita Sarma, James Herbsleb, and Andre von der Hoek. Challenges in measuring, understanding, and achieving socio-technical congruence. In *Socio-Technical Congruence Workshop in conj. ICSE 2008, Leipzig, Germany*, May 2008.

[188] Anita Sarma, Larry Maccherone, Patrick Wagstrom, and James Herblseb. Tesseract: Interactive visual exploration of socio-technical relationships in software development. In *International Conference on Software Engineering Tool Demo*, 2009.

[189] Anita Sarma, David Redmiles, and André van der Hoek. Empirical evidence of the benefits of workspace awareness in software configuration management. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, SIGSOFT '08/FSE-16, pages 113–123, New York, NY, USA, 2008. ACM. Available from: http://doi.acm.org/10.1145/1453101.1453118, doi:http://doi.acm.org/10.1145/1453101.1453118.

[190] Anita Sarma, A. Van Der Hoek, and L.T. Cheng. A need-based collaboration classification framework. In *Proc. on Eclipse as Vehicle for CSCW Research, Workshop at CSCW 2004*, pages 1–5. Citeseer, 2004. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.3622&rep=rep1&type=pdf.

[191] Steve Sawyer. Software development teams. *Commun. ACM*, 47(12):95–99, 2004. doi:http://doi.acm.org/10.1145/1035134.1035140.

[192] Kjeld Schmidt and Carla Simone. Coordination Mechanisms: Toward a Conceptual Foundation of CSCW Systems Design. *Computer Supported Cooperative Work*, 5:155–200, 1996.

[193] Valentin Schöndienst, Spandauer Straß e, Hanna Krasnova, Oliver Günther, and Dirk Riehle. Micro-Blogging Adoption in the Enterprise: An Empirical Analysis. In *10th International Conference on Wirtschaftsinformatik*, number February, 2011.

[194] Adrian Schröter, Thomas Zimmermann, and Andreas Zeller. Predicting component failures at design time. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, ISESE '06, pages 18–27, New York, NY, USA, 2006. ACM. Available from: http://doi.acm.org/10.1145/1159733.1159739, doi:http://doi.acm.org/10.1145/1159733.1159739.

[195] C.B. Seaman and V.R. Basili. Communication and organization: an empirical study of discussion in inspection meetings. *Software Engineering, IEEE Transactions on*, 24(7):559 –572, jul 1998. doi:10.1109/32.708569.

[196] Barbara Senior. Team roles and team performance: Is there 'really' a link? *Journal of Occupational and Organizational Psychology*, 70(1961):241–258, 1997.

[197] J. Shah and C. Breazeal. An Empirical Analysis of Team Coordination Behaviors and Action Planning With Application to Human-Robot Teaming. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 52(2):234–245, April 2010. Available from: http://hfs.sagepub.com/cgi/doi/10.1177/0018720809350882, doi:10.1177/0018720809350882.

[198] Helen Sharp and Hugh Robinson. An ethnographic study of xp practice. *Empirical Softw. Engg.*, 9:353–375, December 2004. Available from: http://portal.acm.org/citation.cfm?id=1017038.1017044, doi:10.1023/B:EMSE.0000039884.79385.54.

[199] Manuel Sosa. A structured approach to predicting and managing technical interactions in software development. *Research in Engineering Design*, 19(1):47–70, 03 2008.

[200] Manuel E. Sosa, Steven D. Eppinger, and Craig M. Rowles. The misalignment of product architecture and organizational structure in complex product development. *Management Science*, 50(12):1674–1689, 2004. doi:http://dx.doi.org/10.1287/mnsc.1040.0289.

[201] Cleidson R. B. Souza and David F. Redmiles. On The Roles of APIs in the Coordination of Collaborative Software Development. *Computer Supported Cooperative Work (CSCW)*, 18(5-6):445–475, September 2009. Available from: http://www.springerlink.com/index/10.1007/s10606-009-9101-3, doi:10.1007/s10606-009-9101-3.

[202] Renée J. Stout, Janis A. Cannon-Bowers, Eduardo Salas, and Dana M. Milanovich. Planning, Shared Mental Models, and Coordinated Performance: An Empirical Link Is Established. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 41(1):61–71, March 1999. Available from:

http://hfs.sagepub.com/cgi/doi/10.1518/001872099779577273, doi:10.1518/001872099779577273.

[203] Lucy Suchman. Constituting shared workspaces. In Yrjö Engeström and David Middleton, editors, *Cognition and Communication at Work*, Cognition and Communication at Work, pages 35–60. Cambridge University Press, 1998.

[204] M Sumner. The impact of electronic mail on managerial and organizational communications. *ACM SIGOIS Bulletin*, pages 96–109, 1988. Available from: http://portal.acm.org/citation.cfm?id=45421.

[205] James Tam and Saul Greenberg. A framework for asynchronous change awareness in collaborative documents and workspaces. *International Journal of Man-Machine Studies*, 64(7):583–598, 2006.

[206] John C Tang, Tara Matthews, Julian Cerruti, Stephen Dill, Eric Wilcox, Jerald Schoudt, Hernan Badenes, and San Jose. Global Differences in Attributes of Email Usage. *Organization*, pages 185–194, 2009.

[207] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *International Conference on Software Engineering, Waikiki, USA*, 2011.

[208] Joshua Tyler, Dennis Wilkinson, and Bernardo Huberman. E-Mail as Spectroscopy: Automated Discovery of Community Structure within Organizations. *The Information Society*, 21(2):143–153, April 2005. Available from: http://www.informaworld.com/openurl?genre=article&doi=10.1080/01972240590925348&magic=crossref||D404A21C5BB053405B1A640AFFD44AE3, doi:10.1080/01972240590925348.

[209] Joshua R. Tyler, Dennis M. Wilkinson, and Bernardo A. Huberman. Email as spectroscopy: automated discovery of community structure within organizations. *The Information Society*, 21(2):143–153, April 2005. Available from: http://portal.acm.org/citation.cfm?id=966268.

[210] J.R. Tyler and J.C. Tang. When can I expect an email response? A study of rhythms in email usage. In *Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, number September, page 258. Kluwer Academic Publishers, 2003. Available from: http://portal.acm.org/citation.cfm?id=1241902.

[211] Giuseppe Valetto, Sunita Chulani, and Clay Williams. Balancing the value and risk of socio-technical congruence. In *Socio-Technical Congruence Workshop in conj. ICSE 2008, Leipzig, Germany*, May 2008.

[212] Giuseppe Valetto, Mary Helander, Kate Ehrlich, Sunita Chulani, Mark Wegman, and Clay Williams. Using software repositories to investigate sociotechnical congruence in development projects. In *Workshop on Mining Software Repositories in conj. ICSE 2007, Minneapolis, USA*, 2007.

[213] Andrew H. Van De Ven, Andre L. Delbecq, and Jr. Richard Koenig. Determinants of coordination modes within organizations. *Americal Sociological Review*, 41(2):322–338, 1976.

[214] GD Venolia and Carman Neustaedter. Understanding sequence and reply relationships within email conversations: a mixed-model visualization. *of the SIGCHI conference on Human*, (5):361, 2003. Available from: `http://portal.acm.org/citation.cfm?id=642674`, `doi:10.1145/642611.642674`.

[215] Fernanda B Viégas, Scott Golder, and Judith Donath. Visualizing Email Content : Portraying Relationships from Conversational Histories. *Interfaces*, pages 979–988, 2006.

[216] Patrick Wagstrom, J.D. Herbsleb, and Kathleen Carley. Decaying sociotechnical congruence as a method to account for architectural changes. In *Workshop on Socio-techical Congruence in conj. Intl. Conf. on Software Engineering, Vancouver, Canada*, 2009.

[217] J. B. Walther. Interpersonal Effects in Computer-Mediated Interaction: A Relational Perspective. *Communication Research*, 19(1):52–90, February 1992. Available from: `http://crx.sagepub.com/cgi/doi/10.1177/009365092019001003`, `doi:10.1177/009365092019001003`.

[218] Diane B. Walz, Joyce J. Elam, and Bill Curtis. Inside a software design team: knowledge acquisition, sharing, and integration. *Commun. ACM*, 36:63–77, October 1993. Available from: `http://doi.acm.org/10.1145/163430.163447`, `doi:http://doi.acm.org/10.1145/163430.163447`.

[219] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, UK, 1994.

[220] Karl E. Weick and Karlene H. Roberts. Collective Mind in Organizations: Heedful Interrelating on Flight Decks. *Administrative Science Quarterly*, 38(3):357, September 1993. Available from: `http://www.jstor.org/stable/2393372?origin=crossref`, `doi:10.2307/2393372`.

[221] Steve Whittaker and Candace Sidner. Email overload : exploring personal information management of email. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, pages 276–283. ACM, 1996. Available from: `http://portal.acm.org/citation.cfm?id=238386.238530`.

[222] Steve Whittaker and Candace Sidner. Email overload: exploring personal information management of email. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 276–283, New York, NY, USA, 1996. ACM Press. Available from: http://portal.acm.org/citation.cfm?id=238530, doi:10.1145/238386.238530.

[223] Timo Wolf, Adrian Schröter, Daniela Damian, and Thanh Nguyen. Predicting build failures using social network analysis on developer communication. In *Intl Conf on Software Engineering, Vancouver, Canada*, pages 1–11, May 2009. doi:http://dx.doi.org/10.1109/ICSE.2009.5070503.

[224] Timo Wolf, Adrian Schröter, Daniela Damian, Lucas D. Panjer, and Thanh H.D. Nguyen. Mining task-based social networks to explore collaboration in software teams. *IEEE Software*, 26(1):58–66, 2009. doi:http://doi.ieeecomputersociety.org/10.1109/MS.2009.16.

[225] Heng-Li Yang and Jih-Hsin Tang. Team structure and team performance in is development: a social network perspective. *Inf. Manage.*, 41:335–349, January 2004. Available from: http://portal.acm.org/citation.cfm?id=972060.972066, doi:10.1016/S0378-7206(03)00078-8.

[226] Yunwen Ye, Kumiyo Nakakoji, and Yasuhiro Yamamoto. Understanding and improving collective attention economy for expertise sharing. In Zohra Bellahsène and Michel Léonard, editors, *Advanced Information Systems Engineering*, volume 5074 of *Lecture Notes in Computer Science*, pages 167–181. Springer Berlin / Heidelberg, 2008.

[227] Robert K. Yin. *Case Study Research: Design and Methods*, volume 5 of *Applied Social Research Methods Series*. SAGE Publications, third edition edition, 2003.

[228] Sergey Zeltyn, Peri Tarr, Murray Cantor, Sateesh Kannegala Robert Delmonico, Mila Keren, Ashok Pon Kumar, and Segev Wasserkrug. Improving efficiency in software maintenance. In *Mining Software Repositories 2011, Honolulu, USA, May 21–22*, 2011.

[229] Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. Predicting defects for eclipse. In *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, PROMISE '07, pages 9–, Washington, DC, USA, 2007. IEEE Computer Society. Available from: http://dx.doi.org/10.1109/PROMISE.2007.10, doi:http://dx.doi.org/10.1109/PROMISE.2007.10.

Appendix A

Ship Team Members

The team members that are directly connected to the Ship Tango project are listed below. Their names have been modified to maintain anonymity. The project experience was acquired from either interviews, in which we either asked the person directly or a team member about that person, or from a questionnaire administered at the beginning of the study (Appendix B).

As noted in Section 4.2.1, the project role is not necessarily stable for the entire duration of the study.

Name	Location	Role	Project experience
Cathy	United States	Portfolio manager	Unknown
Barry	United States	Project manager	Unknown
Mark	United States	Senior Development Leader	7 years
Ann	United States	Development Leader	5 years
John	United States	System Architect	Unknown
Will	United States	Developer	Unknown
Perry	United States	Developer	4–5 years
Nancy	United States	Test leader/Tester	Unknown
Aaron	Brazil	Senior developer	3 years
Yun	Brazil	Developer	1.5 years
Michelle	Brazil	Developer	1.5 years
Lilly	Brazil	Tester/Test leader	1.5 years
Adrian	Brazil	Tester	$<$ 1 year
Sammy	Brazil (contractor)	Tester	1 year
Jason	Brazil (contractor)	Developer	1 year
Jim	Brazil (contractor)	Tester	1 year
Ron	Brazil (contractor)	Developer	1 year
Magda	Brazil (contractor)	Developer	1 year
Kevin	Brazil	Environment coordinator	Unknown
Thomas	United States	Environment coordinator	Unknown
Dan	Brazil (contractor)	Developer	$<$ 1 year

Appendix B

Ship Questionnaire

The questionnaire was designed originally by Sabrina Marczak as a part of her research project [148]. Only data from one page (of seven) was used to provide context for the awareness empirical investigation and the communication pattern study, and consequently I include only the relevant questions of the questionnaire.

Dimension 1 – Demographic data

1. What is your name?

2. How many years of professional experience in Information Technology area do you have?

3. 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

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?
☐ Office ☐ Home ☐ Home and Office ☐ Other:

7. Which type of project is “Outbound Infrastructure Improvements Release 1, Iteration 3” project?
☐ Maintenance ☐ New development ☐ Other:

8. Which is/was your role in the “Outbound Infrastructure Improvements Release 1, Iteration 3” project?
☐ Development Leader
☐ Developer
☐ Product Manager
☐ Project Manager
☐ Requirements Analyst
☐ System Architect
☐ Technical Leader
☐ Test Leader
☐ Tester
☐ Other:

9. When did you join the project? Since:
☐ Envisioning phase
☐ Planning phase
☐ Developing phase
☐ Stabilizing phase

Appendix C

Human Research Ethics Board Approval

This research was submitted to the University of Victoria Human Research Ethics Board under the title, "Investigating Collaboration under Conditions of Change in Global Software Development". The principal investigator was Irwin Hin-Bong Kwan, the co-investigator was Sabrina Marczak, and the supervisor was Daniela Damian. The start date of the research was November 20, 2006 and the end date was November 19, 2009. The research was given Protocol Number 06-317.

The research was approved by Dr. Richard Keeler on November 20, 2006.

Appendix D

Publications From This Dissertation

Portions of this dissertation appear in the following publications.

Referred Journal Publications

Kwan, Irwin; Schröter, Adrian; Damian, Daniela. *Does Socio-Technical Congruence Have An Effect on Software Build Success? A Study of Coordination in a Software Project.* Transactions on Software Engineering, Volume 37 (3): 307–324, 2011. [140]

Refereed Conference Publications

Kwan, Irwin; Damian, Daniela. *The Hidden Experts in Software-Engineering Communication (NIER Track).* International Conference on Software Engineering, 2011. [137]

Refereed Workshop Publications

Kwan, Irwin; Damian, Daniela. *Extending Socio-technical Congruence with Awareness Relationships.* Social Software Engineering 2011, in conj. ESEC/FSE, 2011. [136]

Marczak, Sabrina; Kwan, Irwin; Damian, Daniela. *Investigating Collaboration Driven by Requirements Investigating Collaboration Driven by Requirements in Cross-Functional Software Teams.* Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills, in conj. Requirements Engineering, 2009. [152]

Kwan, Irwin; Schröter, Adrian; Damian, Daniela. *A Weighted Congruence Measure.* Socio-technical Congruence Workshop, in conj. Intl. Conference on Software Engineering, 2009. [139]

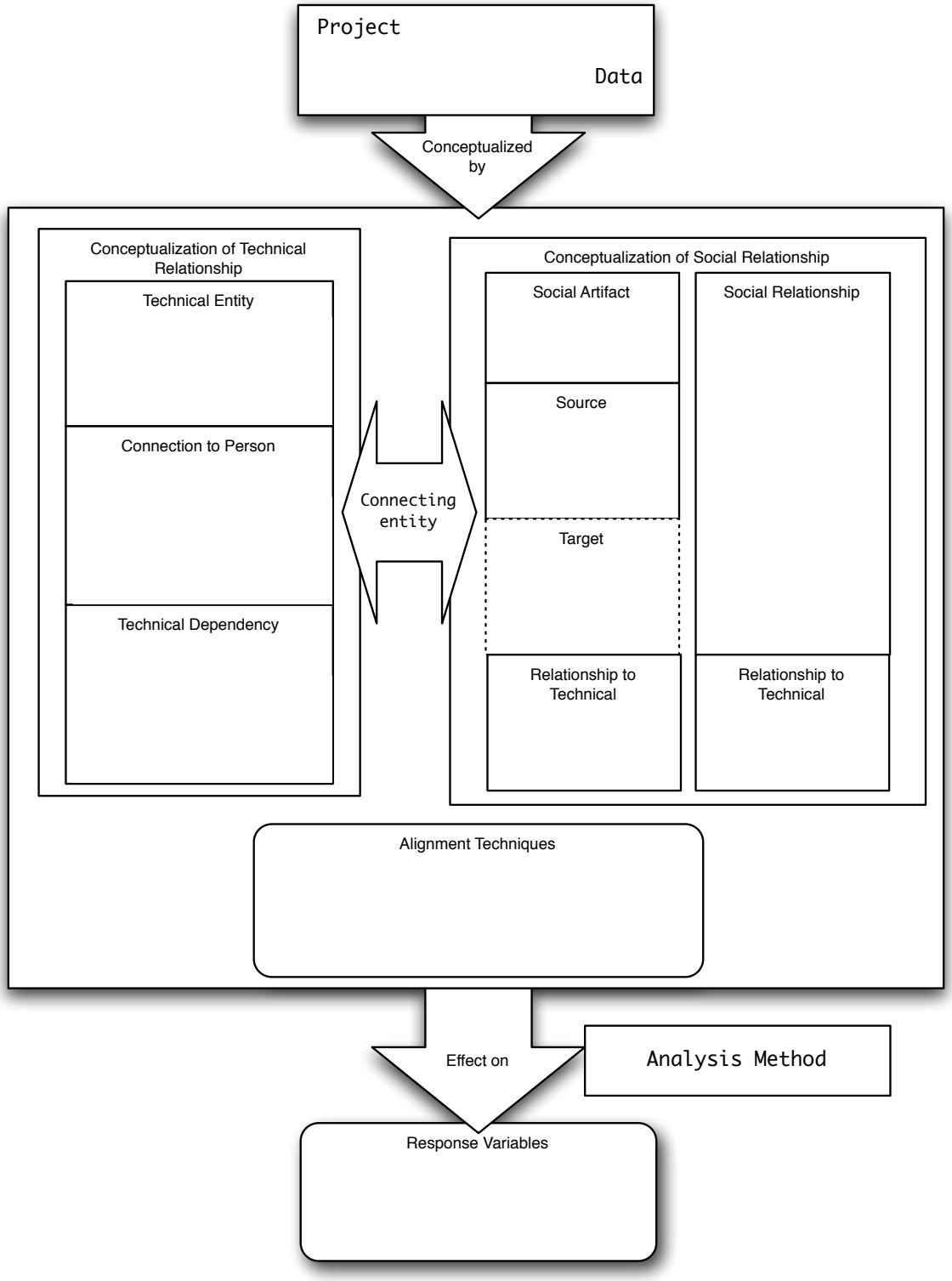
Kwan, Irwin; Marczak, Sabrina; Damian, Daniela. *The Effects of Distance, Experience, and Communication Structure on Requirements Awareness in Two Distributed*

Industrial Software Projects. Global Requirements Engineering Workshop in conj. Requirements Engineering, 2007. [138]

Appendix E

The Socio-technical Congruence Study Template

For convenience, the socio-technical congruence study template can be found on a standalone page for reproduction. A digital version of the source file can be found at `http://www.segal.uvic.ca/projects_public/stc-template.pdf/view` (PDF) and `http://www.segal.uvic.ca/projects_public/stc-template.graffle/view` (Omnigraffle).


Project
Data
Conceptualized by
Conceptualization of Technical Relationship
Technical Entity
Connection to Person
Technical Dependency
Connecting entity
Conceptualization of Social Relationship
Social Artifact
Source
Target
Relationship to Technical
Social Relationship
Relationship to Technical
Alignment Techniques
Effect on
Analysis Method
Response Variables

Glossary

actor In social network analysis, an actor is an entity that has edges. In the context of this dissertation, an actor is a person who can send and receive messages.

alignment technique An element in the socio-technical congruence model that encompasses techniques related to calculating the alignment between technical relationships and social relationships. Alignment techniques in the model include multi-variable modelling, gap sizes, and weighted socio-technical congruence.

awareness Having an understanding of the activities of others that form a context for your own activities.

broker In social network analysis, an intermediary between two groups. In a communication network, a broker is a person in between two other people and usually has power to influence or control the flow of information from one group to another.

build The process of compiling source code into binaries that are consequently linked to form an executable. Also refers to the output of such a process.

centrality A social network analysis measurement that identifies how well-connected a person is to others. Types of centrality include closeness centrality, degree centrality, and eigenvector centrality..

change set In RTC, a set of files submitted by a developer to the source code repository. Every change set is submitted by an author, and is associated to a build and one or more work items.

closeness centrality The average length of the shortest path from this vertex to every other vertex in the network.

comment In RTC, a written contribution by a developer on a work item. Usually, a comment provides additional information about a work item that is not in the work item description.

conceptualization An element in the socio-technical congruence model that encompasses what entities and relationships should be represented in the model to best reflect socio-technical coordination.

connecting entity An entity that identifies the link between the technical relationships and the social relationships.

degree centrality The number of incoming or outgoing edges to a node.

edge In social network analysis, an edge is a connection between two actors. In the context of this dissertation, an edge is a communication instance between one actor and another.

eigenvector centrality The level of connectedness a vertex has to other well-connected vertices.

emergent person A person included in an online discussion only after an initial set of recipients has been contacted.

emergent replier An emergent person who replies to a messages in the thread and thus becomes actively involved in the discussion.

explicit communication Information intentionally sent to others.

gap An instance in socio-technical congruence where a technical relationship between two people exists, but no social relationship exists between the same two people.

gap size The difference in the number of technical relationships and social relationships in socio-technical congruence. A large gap size suggests that few social relationships satisfy the existing technical relationships, whereas a small gap size suggests that many social relationships satisfy the technical relationships.

Iimportant communicator A person who is heavily involved in work-related communication sent throughout a team.

initial recipients The people who receive the first message of the thread. These recipients are explicitly added by the initial sender.

initial sender The person who sends the first message of a thread..

Management information systems (MIS) A system that provides information needed to manage organizations effectively.

multi-variable socio-technical congruence An extension of socio-technical congruence that allows the combination of different relationship networks to analyze alignment.

Rational Team Concert (RTC) A product developed by IBM that integrates multiple aspects of development, including source-code control, issue tracking, and the integrated development environment into a single suite of tools.

Ship The pseudonym for a shipping system being developed by one of the teams investigated in this dissertation.

situation modelling An element in the socio-technical congruence model that encompasses exceptional and interesting situations that should be representable using a socio-technical congruence conceptualization. Situations include emergent people and the representation of managers, supporting staff, and customers.

social actor A person or organization in a social network. Represented as an actor.

social artifact An artifact created as a byproduct of a social relationship, such as an email message.

social network A structure that represents relationships between people. A social network can be represented as a graph G such that $G = V, E$ where V is a set of vertices and E is a set of edges which are two-element subsets of V.

social network analysis (SNA) A quantitative analysis technique that represents actors and the connections between actors as a network, and identifies the properties of such a network.

social relationship A relationship between two team members that represents coordination that happens in reality. Examples of a social relationship is when one person sends an email to another.

social relationship network A social network where the relationships between the actors in the network are social relationships.

socio-technical congruence A measurement of alignment between social relationships and technical relationships. If a social relationship exists where there also exists a technical relationship, then the relationship is *satisfied*. The socio-technical congruence index is calculated as the number of satisfied relationships over the total number of technical relationships.

socio-technical congruence model A representation of socio-technical congruence that describes, in general, how socio-technical congruence should be conceptualized in order to study socio-technical coordination in an organization.

team mental model The organized mental representations of the key elements within a team's relevant environment that are shared across team members.

technical assignment A relationship between a technical entity and a person.

technical dependency A type of dependency between two technical entities.

technical entity An entity in a project that can be worked on by a person or a group. Examples of a technical entity include a source code file, a compiled binary, a requirement, a task, or a bug.

technical relationship A relationship that indicates that one person is dependent on another person. This relationship is calculated based on a person's work assignments and work dependencies.

technical relationship network A social network where the relationships between the actors in the network are technical relationships.

thread An aggregation of email messages with the same subject title. Generally, an email thread contains messages that are replies to a previous message..

weighted socio-technical congruence A representation of socio-technical congruence where each social relationship in a social relationship network and each technical relationship in a technical relationship network has a value. Unlike in unweighted socio-technical congruence, the values can be values other than 0 or 1.

work item In RTC, a description of an issue. Usually, a work item describes a defect or a feature, and is repaired by a change set..