Communication in Software Engineering Teams

by

Thanh H.D. Nguyen
B.Sc., University of Victoria, 2007

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

MASTER OF SCIENCE

in the Department of Computer Science

Your file Votre référence
ISBN: 978-0-494-66817-7
Our file Notre référence
ISBN: 978-0-494-66817-7

Supervisory Committee

Communication in Software Engineering Teams

by

Thanh H. Nguyen
B.Sc., University of Victoria, 2007

Dr. Daniela Damian, Department of Computer Science
Co-Supervisor

Dr. Ulrike Stege, Department of Computer Science
Co-Supervisor

Abstract

Supervisory Committee
Dr. Daniela Damian, Department of Computer Science
Co-Supervisor
Dr. Ulrike Stege, Department of Computer Science
Co-Supervisor

Communication is an important activity within software engineering teams as within any other type of organization. In distributed setting, distance has been reported to introduce significant delay in communication. However, new processes and tools have been specifically introduced to alleviate the effect of distance on distributed development. In order to examine if the new processes and tools have indeed made a different on distributed development, we conduct an empirical study in communication of a large globally distributed software engineering team. The goals of our study are to a) investigate the effects of distance on communication speed and b) examine the structure of communication network of this team. We found that distribution does not affect communication speed as reported in previous studies. We also found that this team was able to maintain a project wide communication network with a large core of contributors from across different sites. We conjectured that this structure of communication network helps teams to overcome the challenges of distribution. Finally, we explain the implications of our findings to practitioners and suggest directions for future studies.

Table of Contents

List of Tables

List of Figures

Acknowledgments

Firstly, I would like to thanks my two supervisors, Dr. Daniela Damian and Dr. Ulrike Stege for teaching me how to conduct a proper scientific study, write proper English, and for the financial support during my study. I do not think I can finish the study without either one of you.

I would also like to thank the members of the SEGAL lab, Sabrina Marczak, Irwin Kwan, Adrian Schroeter, Lucas Panjer, and Timo Wolf, who gave me constants feedback with the thesis and helped me with the writing. I especially grateful for the help of Adrian, Lucas, and Timo, who helped me with the data collection process.

I would also like to thank Harold Ossher and Jean-Michel Lemieux at IBM who helped me collect the IBM Jazz Development data.

I also thank my cousin Tri Huynh and my friend Eoin Whitney for helping me editing the last version of this thesis.

Lastly, I would like to thank my parents for their unconditional love and support.

Chapter 1: Introduction

In recent years, *globally distributed development* (GSD) has become an industry trend [5, 49]. Of the US Fortune 500 companies, 203 are involved in offshore outsourcing [2]. Software companies which adopted GSD benefit from many advantages, such as reduced production cost, accessibility to a highly-skilled labour market, and a reduction of the distance to customers. However, distributed development introduces many challenges to software development teams. Although the development of communication tools such as instant messenger, email and audio video conferencing has enabled software developers to work remotely, globally distributed development teams continue to experience coordination problems due to the distance between team members. In this thesis, we explore some of the current problems that trouble GSD teams. In particular, we investigate the possible delay in communication and the possible increased difficulty in coordination.

Literature on GSD reports that distributed teams often report coordination problems [20, 34, 10]. *Coordination problems* refer to difficulties in the process of integrating each team member's activities into the contributions towards the common goals. Such problems include dividing tasks, setting deadlines, arranging meetings or reporting progress. Some of the problems reported include breakdowns in coordination [18, 6], the lack of a common (formal or informal) communication channel [17, 47] or mismatches between the required and actual coordination [13]. While coordination problems are inherent aspects of any large organization, the nature of software development makes them inevitable. For example, developers from different sites can miss an important milestone because they may be unaware of a requirement change due to an overload of information from the team mailing list [18]. Team members might not be able to access valuable information from another team because of the lack of informal communication with other teams [47]. In distributed settings the problems seem to be more challenging than in non-distributed ones, due to various reasons [35, 6, 39, 9].

Our main assumption in this thesis is that communication is the process that underlines the coordination process. Through effective communication, team members are able to coordinate their activities and establish common goals, policies, standards, and quality level. Previous studies found that communication in a globally distributed development has suffered from many challenges such as the loss of “communication richness” due to geographic distance [6], misunderstandings caused by cultural differences [53], or delay in communication [34]. During our study, we have come to believe that the communication problems in GSD teams are the main causes for coordination problems. Improved communication processes in GSD teams will enable the teams to improve their coordination processes. This will result in improved productivity, lower cost, and better software. The potential benefits motivate us to start looking into the communication of globally distributed teams.

Our motivation to study communication in GSD teams is also supported by the field of organizational studies. The importance of the communication process has been emphasized and studied in organizational studies of the past century. For example, past studies looked at (a) patterns of communication among marketing, engineering and manufacturing teams [29], (b) the reasons why there are problems interfacing research development teams with marketing teams [31], and (c) the effect of the distribution of knowledge on group performance [56]. As software teams can be considered organizations, this knowledge should be applicable to communication in GSD teams as well. In fact, the awareness of communication problems in distributed software teams has improved in the last few years. Guidelines have been developed for project managers in distributed settings [10, 49]. Further, tools have been developed to integrate communication support into the working environments [48]. However, we do not know if the new guidelines and tool support really help improve communication nor do we know how exactly the communication structure of distributed team should be. Our research aims to find the answers to these questions. We summarize our problem statement below.

Summary of the problem statement

Communication is a very important aspect of global software development. Studies in the past reported that distributed teams suffered from communication problems due to the affects of distance. Although new guidelines and tools have been produced to alleviate communication problems, we do not know if they really help improve communication in distributed teams nor do we know how exactly the communication structure of distributed team should be.

Research goals

We decided to conduct a case study on communication in a large GSD team, called the *IBM Jazz Development team*, with the aim that this case study will give us a better insight into the current state of practice in communication of GSD teams.

We focus on two aspects of communication:

- *Effect of distance on communication speed*: Delay in communication has been a major challenge of many software managers when dealing with distributed teams or customers [37]. Past investigations showed that distributed communication can introduce on average a day and a half delay compared to collocated communication [34]. Cause of the communication delay seems to be the geographic distance.

  Our first goal is to re-examine whether recent improvements in software engineering practice and tool support have improved this problem.
- *Structure of the communication network*: Communication connects people. Social network analysis has been used to study communication patterns in sociology and organization studies. For example, study of an automotive engineering team' communication network reveals the advantage of certain engineering processes [29]. A research conducted by Hinds and McGrath [38] found that certain structure of the communication network facilitates an improved coordination in globally distributed teams.

  Our second goal is to examine the structure of our case study's underlying communication network.

Research methodology

To address the problem described in this thesis, we conducted a case study of communication in a large globally distributed software engineering team: the IBM Jazz Development team [41]. The team consists of 151 members and is distributed across 16 different sites in Canada, USA, and Europe. Within the global team, there are 47 functional teams. The company uses a centralized work tracking system called the *work* *item repository*. Every team member uses this system as their main communication channel. This made our study of the team communication possible.

We extracted the team's communication activities from the work items repository. The work items are the main unit of analysis. Each work item contains comments from different developers working on the work item. Each work item also has attributes such as response time, resolution time, location and time zone of developers. To determine the effect of distance on communication, we extracted the work items' attributes (e.g., response time and number of location) and applied statistical tests to find possible correlations among them. To study the team's underlying communication network, we linked the developers through their comments in the work item repository. We then applied techniques from social network analysis to determine the core-periphery property of this network. To determine the motivation and the cognitive orientation of the communication content, we extracted comments from the work items' repository. Then, we perform different content analyses on the comments.

Research contribution

Our overall goal is to study and understand communication in GSD research. As a very first step of this process, our findings from this case study aim to provide better understanding about communication in distributed teams for both researchers and practitioners.

To software engineering researchers, our study adds to the existing knowledge about communication in distributed software engineering teams. First, we provide evidence that the effect of the distance on speed of communication is no longer as strong as what had been reported [34]. We conjecture that this advance in distributed communication is the result of improved processes and tools as we observed in our case study. The Jazz team is very experienced in distributed development. Their processes, called the Eclipse Way [27], are specifically designed to facilitate globally distributed development. More importantly, the team uses the Jazz Platform. Which is a collaborative development tool that was specifically built to bring communication and collaboration support into the development environment. Secondly, we bring insights about this global team's communication network that may explain the lesser effect of distance on its communication. In particular, we found that the communication network of the Jazz team has a hierarchical structure although the communication was carried out organically without such a defined structure. According to previous research [38], this property is beneficial for coordination of team activities. Our study also suggests different directions for future research on the communication of software teams.

To practitioners, our study answers some questions about their practice. First, we found that practitioners should try to use communication speed to improve their software team's productivity. Secondly, we provide evidence that practitioners should not be afraid of distributed development as long as they have the right tools and processes to overcome the distribution factor of the teams.

Thesis organization

The purpose of this chapter is to provide the reader the context of our study, as well as our motivation to study communication in GSD teams. In Chapter 2, we provide an overall review of existing literature about communication in organizational studies and software engineering. These studies, although not considering communication as a topic on its own, provide the motivation for our research.

In Chapter 3, we describe our study settings, the IBM Jazz team, followed by our methods of data collection and analysis. We also explain the data constructs that we use in the subsequent chapters where we examine our research questions.

In Chapter 4, we (re)examine whether recent improvements in software engineering practice and tool support have eliminate the effect of distance on communication as documented in previous work [34].

Chapter 5 examines the core-periphery property of the social networks built on the team’s underlying communication structure.

Finally, we discuss the overall results in Chapter 6 and suggest future works in distributed communication for GSD teams.

Chapter 2: Background and Related Work

Challenges in global software engineering

In software engineering, our literature search first focuses on studies involving global software development teams. In recent years, globally distributed development has become an industry trend [5, 49]. Of the US Fortune 500 companies, 203 are involved in offshore outsourcing [2]. In 2009, Gartner Research reported in their Market Trends: Application Development, Worldwide, 2008-2013 [55] that the projected total revenue growth for outsourcing application testing alone is an estimated $2.4 billion in 2009. The projected growth is 17% from 2007 through 2012.

Software companies which adopted GSD benefit from many advantages, such as reduction of production cost, accessibility to a highly-skilled labour market, quick turnaround in forming teams to exploit market demand, "round the clock" development and a reduction of the distance to customers [36]. However, distributed development introduces many challenges to software development teams. Although the development of communication tools such as instant messenger, email and audio video conferencing has enabled software developers to work remotely, globally distributed development teams (still) experience coordination problems. Their problems can be categorized into six main themes according to Herbsleb and Moitra [36]:

- *Strategic issues*: dividing up the work across sites is difficult, resistance to GSD is often high at well established sites.

- *Cultural issues*: serious misunderstandings due to cultural differences occur, cultural differences often lead to communication problems.
- *Inadequate communication*: important informal communication is missing due to geographical distance, restricted and filtered communication is often required because of intellectual property concerns.
- *Knowledge management*: sharing information between customers and developers is difficult, determining critical tasks among different sites is also challenging.
- *Project and process management issues*: synchronising processes status across different geographic locations is critical and often difficult.
- *Technical issues*: global network connections are often slow and unreliable, tools are hard to maintain across organization boundaries.

In this thesis, we emphasise our inquiry into the communication problems that trouble GSD teams which belong to the third theme on the list, namely inadequate communication. In particular, we investigate a possible delay in communication and a possibly increased difficulty in coordination.

Distance and delay in communication

We examine the early work of French and Layzell [25] who interviewed people from five different organizations about their experience with distributed software development. Recorded communication problems include a high time investment for communication, a lack of understanding between different sites, and a too high reliance on expertise of colleagues working in remote sites. These problems are echoed in later reports such as the ones from Herbsleb and Grinter [30] or Battin and Crocker [6]. These reports further discussed counter measures to the distance challenges in GSD, such as a centralized bug report system. The reports recommend to avoid imposing a common process and instead to encourage each site to have their own [6].

To study the effects of distance on communication speed in GSD teams, we focused our literature search on studies that deal with how distance affects delay in GSD. We found a series of empirical investigations of distance and delay in software development by Herbsleb and colleagues [30, 32-35, 37].

In early studies, Herbsleb and colleagues reported about communication problems and lengthened cycle times to resolve systems issues (e.g., [32, 33]). They then followed up by systematic studies on the correlation of distance and speed in large distributed organizations (e.g., [34, 37]). The 2003 study at Lucent [34] provides systematic evidence about a significant communication delay and a task completion delay for modification requests involving cross-site work. Here, a comparison of data from same-site and cross-site projects indicates that tasks involving distributed participants take about 2.5-times longer to complete than similar collocated tasks. This result is explained by the perceived communication delay as reported in interview data. Other factors influencing task completion time included the number of people involved in the task, as well as the size of the task. Not surprising to this study, there were significant differences in the size of distributed vs. same-site communication networks. Negative impact of distance on the properties of distributed social networks has recently been confirmed by Ehlrich and colleagues [22].

In support of the findings mentioned above, further studies have been conducted. An experience report of nine distributed projects at Siemens [37] brings insights about benefits and challenges in distributed work, identified through interviews conducted at three different sites. Besides reported benefits, communication and collaboration across sites continued to be identified as a big problem, leading to a reduced development pace. The interviewees clearly stated that face-to-face communication is their perceived most important and fastest way of communication. They also reported that frustrations arose when the pace of interaction declined.

This series of work by Herbsleb and colleagues [32-35, 37] forms much of our understanding about speed and communication in distributed teams. However, companies have been revising their communication practices to cope with distributed development in the past few years. Many tools have also been designed to assist distributed development. In [10], Carmel and Agarwal suggest tactics to alleviate the impact of distance. These recommendations include: reduce the number of tasks that require intensive collaboration, reduce cultural distance by enforcing a common organization culture, and reduce temporal distance by utilising asynchronous communication channels. Lings et al. [49] also recommended ten strategies for successful development. They further outlined how to use these strategies to tackle specific process and difficulty in distributed development. On the tool support side, commercial [45, 41] and open source / academic tools [16, 43] have been developed to facilitate distributed development. However, we do not know if the new guidelines and tool support really help improve communication nor do we know how exactly the communication structure of distributed team should be. That is why, we decided to investigate the IBM Jazz Team, a large distributed team. The team uses their own product, the IBM Rational Jazz Platform [41] which aims to assist communication and coordination in large distributed team. They are also very experienced with distribute development because of their previous product (more details is in the next chapter). They are a good candidate to re-examine the industry's state of practice in distributed communication.

The use of social networks in the study of developer communication

Social network analysis is a powerful analysis tool to investigate relationships between individuals. When two developers exchange conversations regularly, regardless of the medium (e.g., face-to-face, online messaging, or work items' comments), they establish a stronger working relationship to each other than to those in the team they are not communicating with. Through communication, consciously or unconsciously, they become aware of each other's activities, exchange technical expertise, influence each other's decision, and ultimately coordinate their activities. The field of social network analysis established methods to study this relationship between different members of the team [52, 46].

We use a running example to illustrate some of the terminology and techniques in social network analysis. As an example consider the following situation. Jana, Jeff, Joanna, and Jason are four members of a testing team. The team is distributed into two different locations. Jana and Jeff are in Victoria and Joanna and Jason are in Ottawa. The number of times the four team members communicated during the development of a feature is recorded in the following table.

Table 1: Number of times team members communicated during the development process

	Jana	**Jeff**	**Joanna**	**Jason**
Jana		37	17	4
Jeff			23	7
Joanna				25

In this hypothetical example, without knowing the team's work structure, we can determine who is more senior in the company and holds more responsibility in the development process. Observing the communication network, we can see that Jeff communicated more with the other development site: Jeff communicated 30 times with Joanna and Jason compared to the 21 of Jana. The same holds for Joanna on the other site. This can be interpreted as follows: Jeff and Joanna are more senior; they hold a more managerial role compared to Jana and Jason. This requires both to communicate more with other teams. Using social network analysis, we can formalize this analysis by using the concept of *degree centrality*. First we construct the social network based on the communication time depicted in Figure 1. In graph-theoretic terminology, Table 1 can be viewed as an *adjacency matrix* where each name of a row (or column) represents a *node* in the graph. In this case, each node corresponds to a team member. The graph has *edges* for every cell in the matrix that contains a value. Further, the edges are weighted by their corresponding cell values. In our example, the edge weights correspond to the number of times the connected pair of team members communicated. A node's *degree* in the defined weighted graph can be defined as the *total weight* of its adjacent edges. In social network analysis, the node degree is called the *degree centrality* of the node [65]. Considering again our example, the graph's degree centralities are: 58 for Jana, 67 for Jeff, 65 for Joanna, and 36 for Jason. We can see that Jeff and Joanna have the two highest degree centralities in the graph. This means that most of the communication in this test team flows through Jeff and Joanna. In social network analysis, people who have a high degree centrality are suggested to be the *maintenance of communication* or the *coordinators* of the group process [24, 14, 59]. Those are people who have the most experience and thus can redirect inquiries from the outside to other people inside the group. Using this information, we suggest that Jeff and Joanna are the leaders of this test team: both work on two different sites, and therefore they are the leaders the two sites, each coordinating the activities at their own site.

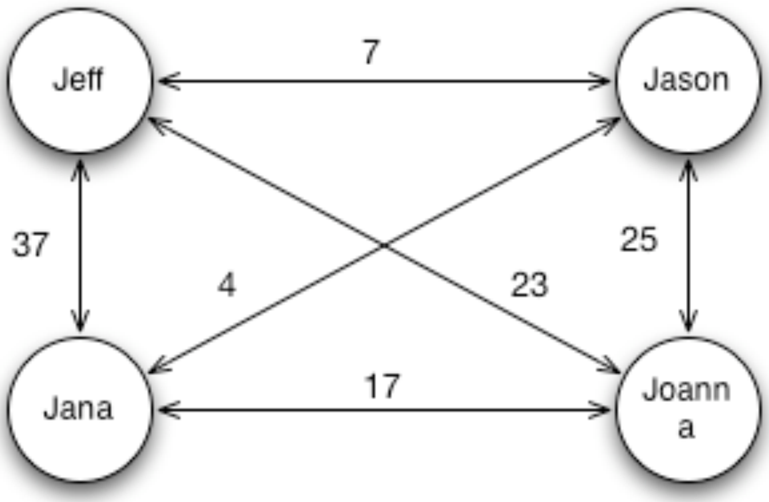

Figure 1: A social network built on the communication data for our hypothetical example

The above example illustrates how the concept of a social network can be used to analyze the property of a group of people. Social network analysis has been used in the area of social studies [40, 7, 11, 44]. With much more complex analyses than the one shown above, social network analysis has been used to understand the working relationships between teams and between team members in organizational studies. For example, Griffin and Hauser [29] constructed a team-to-team communication network to study the effect of two different research and development processes (Quality Function Deployment, and phase-review) on communication. The study discovered that Quality Function Deployment encourages a stronger communication flow between team members at the same team level than an up-over-down flow as observed in the phase-review. This explains why Quality Function Deployment has become the preferred process in comparison to phase review among car manufactures. In another study, Gloor et al. [28] visualized the social networks of different World Wide Web Consortium working groups. They showed that there are significant variations between different groups in terms of the communication patterns and network structures of the different groups. Using the communication network, the authors also found emerging group leaders who were not formally appointed. In a study of research and development teams of a multi-national company, Hinds and McGrath constructed social networks for different teams. They calculated different properties of the networks and correlated them with the ease of coordination reported by the participants. Their result suggests that a hierarchical structure may support the ability to coordinate activities.

In software engineering, many researchers have begun to use social networks to analyse relationships among software team members. Representations such as social networks allow us to capture information about the real world relationships that form among developers whose work is related to each other [19]. A series of papers published by Cataldo and colleagues [13, 12] proposes the use of social networks based on actual communication and task dependency to calculate social technical congruence measures, as follows. They created a social network of the team members based on actual communication by connecting people who commented on the same modification request or chatted about the same modification request during the development process. Then they built a different network of the same team members based on their task allocation by connecting people who were supposed to work together on the same task. From the two networks, they calculated an index called *social technical congruence index*. This measure can be used to indicate the degree to which the actual social structure of the software team (the first network) matches the coordination requirements (the second network) of the software built, as suggested by Conway Law [15]. Marczak et al. [50] make use of social networks in the context of requirements engineering to study the information flow among software developers working on interdependent requirements. Using a flow network, the authors were able to identify key brokers of information among the developers. These people hold important roles because they control the information flow from the dependee network to the dependent network.

Although social network analysis has led to many possibilities in analysing the communication network in global software development team, not everything is known yet in how to interpret the different network measurements in software engineering teams. A lot of the measurements that have been used extensively in social sciences are not easy to generalize for software engineering teams. For example, we know from social studies that people with a high-degree centrally in their acquaintanceship network are normally influential in their community. If we find a software developer with a high degree of centrality in their communication network, as Jeff and Joanna in our hypothetical example, does this mean that they are highly influential in their workplace? The problem is that in a social study, the connections are typically acquaintanceship, friendship, or kinship. These relations have been studied extensively in the past. Working relations among software developers are, on other hand, relatively new in this research area and have not been fully understood. Therefore it is difficult to interpret network measurements of a built network.

What does stand out for us is a study by Hinds and McGrath [38] where 33 distributed and co-located research and development teams were surveyed. One of their reported results is that the teams perform better in terms of coordination when they have an informal hierarchical communication structure. Therefore we decided to build the communication-based social network of this team and determine if the network possesses such a core-periphery structure. We also use new network structures, called *group-degree centrality* and *group efficiency*, as proposed by Everett and Borgatti [23] to measure how connected members are in a geographical location to other project members in the IBM Jazz Team. We may be able to use these measurements to indicate how good or bad the communication structure of a distributed team is. This can help project managers to monitor the health of their distributed communication processes.

In summary, research in global software engineering have identified problems with communication in distributed development team [36]. In particular, studies [30, 32-35, 37] found that communication delay in distributed team is very high. However, in recent years, reports [49, 10] have suggested strategies to improve communication in distributed environment. Many collaborative tools [45, 41, 16, 43] to support distributed communication has also been introduced. As a result, we do not know how the new practices and tools affect communication in recent distributed teams. In this study, we re-examine the effect of distance on communication by conducting a case study at the IBM Jazz development team. We also investigate the communication structure of the team using techniques borrowed from social network analysis in order to gain insights into the results of effects of distance on the communication of the global team.

Chapter 3: Research Methodology

A case study

In early 2007, after we decided to study communication in software engineering, we decided on the research methodology to follow. First, we decided on an empirical study. It is our belief that software engineering research, especially those that involve tools and methods, should be based on empirical data. This is the key to understand the specific factors behind tools and methods success [54]. The second decision is the research population and the scope of the study: an industrial-wide survey or a case study? The first option is to conduct an industrial-wide survey similar to some of the studies [38, 25, 30] mentioned in the above section. A research-wide survey has a high generalizability which means that our findings are more applicable for software teams. However, such a survey is costly and not suitable for looking deep into a topic. As the scope of this study did not allow us to conduct such a survey with the Jazz team and we also felt that communication is a topic that should be explored with depth, we decided to conduct a case study.

Study setting

We picked a company that (a) is a large globally distributed team and (b) has archived their communication during their development process: the IBM Jazz development team. The IBM Jazz development team is a large global software engineering team with development labs in North America and Europe. At the time of our investigation, the team has around 151 members in 16 different locations as shown in Figure 2. For them, ensuring effective communication between the team members is very important. This made Jazz a very suitable candidate for our case study. The goal of the IBM Jazz team is to produce the IBM Jazz platform. The Jazz platform aims to integrate collaboration support into integrated development environments (IDEs) such as Eclipse or Visual Studio [41]. It is one of the newest commercial communications support tools in software engineering. The team host the product during the development process. This makes them an ideal candidate to study the effect of new communication practices and tools in GSD.

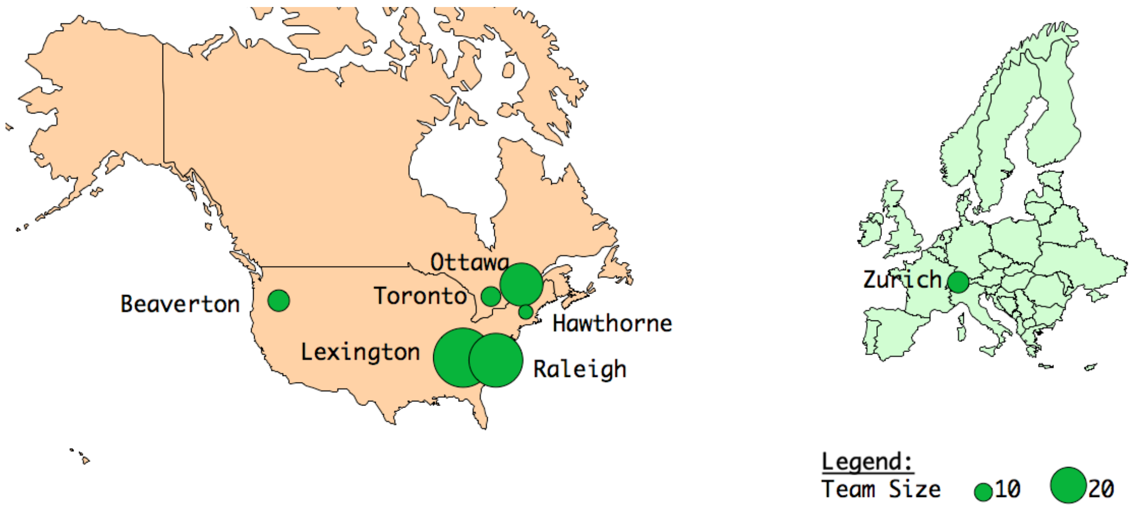

Figure 2: The IBM Jazz Team during the time of this study. The size of the circles indicate the number of members on each site.

Jazz brings support for collaboration into the IDEs such as Eclipse and Visual Studio by providing a central repository to store all the software activities' artefacts such as work items, comments, change sets, and builds. Not only all of the artefacts produced by the platform are saved in a repository, they are linked together in the repository. The team hosts its own product and uses it as its main communication channel between the team members across the globe. This means the communication data is present in the repository and linked to the artefacts. This provides a great opportunity to study team-communication behaviour.

At the time we collected the data from Jazz (May 2007 to April 2008), the Jazz development team consisted of 151 members. The team was distributed across 16 different locations in Canada, USA, and Europe (Figure 2) at the time of our data collection started. Each member belonged to one or several of the 47 teams which were responsible for a certain component in Jazz. We call these teams *component teams* to be distinguished from the main Jazz team. Each component team is led by a team leader. Out of the 16 different locations, there are seven main development sites. Table 2 shows these development sites and these main component teams located at the site. Each component team is responsible for a component in Jazz. We have to note that the different teams and the components evolved over the time of our study. Figure 3 depicts the components and their relations. Some of the components in Figure 3 do not have a corresponding component team although they existed on documentation [42]. We believe that these components were in planning stage or part of another component.

Table 2 Jazz development team's main locations and functions

Location	**Function teams**	**Number of Contributors**
Beaverton	Team Build, Process	12
Hawthorne	Requirement	8
Lexington	Source control, System Test, Requirements, User Assistance	33
Ottawa	Source control	24
Raleigh	Repository, User Assistance	30
Toronto	UI and Dash Board	11
Zurich	Work Item, Iteration Planning	12

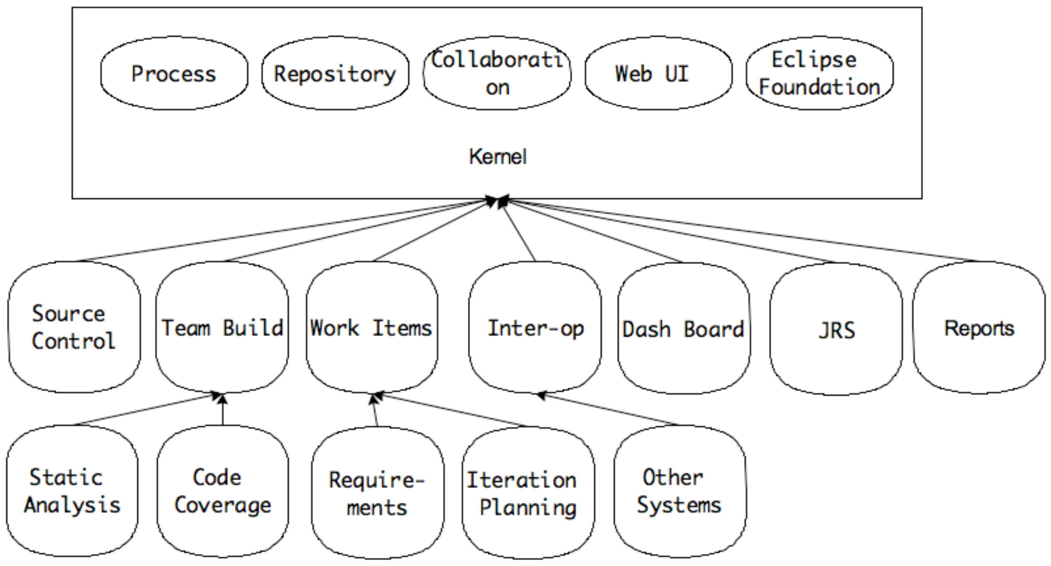

Figure 3 Jazz components and their relations

The origin of the Jazz project is very similar to that of the Eclipse project. The Eclipse project, which goal is to build an open source IDE, is a very successful project led by IBM. IBM started the development of Eclipse in 1998. In 2001, IBM made Eclipse an open source project. This move made Eclipse a successful IDE with a large user base. In 2004, an independent foundation, the Eclipse Foundation, was created to overlook the development of Eclipse. IBM, however, is still the biggest contributor to Eclipse's code base. Although they have to pay for most of the development in Eclipse, which is open source, IBM benefits from Eclipse by building their propriety products on a stable community based platform, e.g. Lotus Notes and WebSphere. The Jazz project follows a very similar business plan. The first release of Jazz was in June 2008. Although Jazz is not an open source project (yet), it is free for not-for-profit use. The code is available for download.

Because of their similar origin, the Jazz team also adheres to the Eclipse Way process [26, 42], a process that was adopted by the Eclipse development team [21]. Most of the people in Jazz have been involved with Eclipse development in the past. At the Ottawa lab, the Jazz and the Eclipse team share the same building. The Eclipse Way process adopted the principles of agile approaches. Each development-iteration is a six weeks development cycle that consists of planning, executing, testing, and retrospection. There is a *project management committee* (PMC) consisting of team leaders and senior engineers. The PMC is responsible for the planning of each release. The team leaders are then responsible for each iteration and oversee the execution of the plan. After the iteration plan has been approved, each functional team is set to work according to the plan.

The Jazz team has three different communication channels aside the regular face-to-face meetings and face-to-face informal communication: the work item repository, the mailing list, and the chat system. The mailing list is mostly used for announcements. The chat system is more for synchronous (real-time) communication. The majority of the communication is going through the work item repository which is our data source. The work item repository is a bug/task tracking system similar to the open source Bugzilla system [61]. As in other bug tracking systems, a work item can be of any of the type defined by one of the development teams, e.g. Story, Track Build Item, Plan Item, Retrospective, Defect, Enhance and Task. After a work item is created, everyone can communicate about the work item by leaving comments. Typically, this will be the communication between the developer who is responsible for the work items and other stake holders such as team leaders, clients or other developers.

Data collection methods

All the information about work items is saved in a central repository which is a web application server with a relational database backend. To collect data for our study, we planned to mine the backend database. However, because of privacy and intellectual property concerns, we did not have direct access to the relational database backend. Instead, we mined the information through the Jazz's Application Programming Interface. We developed a Jazz plug-in that queries the database and exports the data into XML files [51]. Our liaison at IBM's Watson Research Center ran the queries and sent us XML output files. Then, we imported the data into our own MySQL database [1]. This database has a very similar schema to the Jazz backend database. The data analysis is then performed on this database. Because the artefacts and their links span over different parts in the repository, each time we receive a set of artefacts, we may have to fetch other related artefacts. Therefore we have to go through this process several times until we receive every dataset needed [51].

Using the process described above, we extracted a total number of 18,618 work items. After the data were collected, we ran queries on our MySQL-database to check the validity of our data. Because the database schema changed and expanded after each development circle of Jazz to accommodate the new features, some of the data points have missing fields. We noticed that some geographically locations of Jazz team members were unknown. Therefore, the work items that these members were involved in are invalid because most of the analyses in this study require member locations. In turn we had to remove 4,876 such work items from the dataset. After this process we were left with a total of 13,742 work items that were created between October 2005 and November

2007. 0.73% of these work items was created in 2005, 29.46% in 2006, and 69.82% in 2007. The work items have 43,967 different comments in total.

In addition to the information we retrieved from the repository, we acquired the Jazz source code from Jazz.net website and collected geographical data from our contact in IBM. We further asked the developers and managers about different aspect of the process. These exchanges helped us better understand the relationship between the artefacts in the repository and the team practices.

Data construct

The data consisted of artefacts such as work items, requirements, built definitions and results, contributors, and links between them. For our analyses in the following chapters, we only use the information related to the work items. These work item's properties are our constructs.

As an example, we show here an actual work item from our database. Work item number 17170 was created on February 8th, 2007 at 10:52:48 and completed on February 12th, 2007 at 09:09:43. All times denote the server's times. The server is located in Ontario, Canada. Table 3 shows the comments left on the work item. We changed the developers names to protect their identity. We will use this example to demonstrate how we calculate the different data constructs below.

Table 3: Example of work item's comments

Developer (location - team)	**Comment**	**Date**
Carl (Zurich – Work item)	The work item's project area is missing/deleted.	2007-02-09 00:15:12
Zoel (Zurich –	I am catching these and log them, but the indexing itself should not be affected by it in general. Matt, can you	2007-02-09 00:57:16

Work item)	confirm that the exception was a log entry?	
Landner (Raleigh - Repository)	The exception does not look like a log entry, but the indexing does continue after the exception is written. When running a migration the exception is written to standard error and fills the bottom 2331 lines (or 91%) of the logging output and it makes it hard to see immediately see if the migration failed.	2007-02-09 05:10:13
Zoel (Zurich)	I have found the e.printStackTrace offenders and replaced them by slimmer log entries. The stack trace is only logged when a TeamRepositoryException other than the expected ItemNotFoundException is encountered. The indexing in general isn't (and wasn't) affected by these deleted project areas.	2007-02-12 09:12:19

For our analysis of speed and communication in Chapter 4 we use the following data constructs:

- *Response Time*: Response time is an indicator of how fast the communication was carried out for a specific work item. We conceptualize response time by the average delay of the comments starting at the first comment. In our example above, the response time is (0.029 + 0.175 + 3.168)/3 = 3.372 days.
- *Resolution Time*: Resolution time is an indicator of how fast a work item was finished. We chose the time between the creation of the work item and the time it was marked as resolved to conceptualize resolution time. In our example, the resolution time is 3.928 days.
- *Number of sites*: Num of sites is an indicator of the degree of distribution of a work item. For this measurement, we use the number of locations of the commenters. For example, if a work item is commented on by three different developers, two of which are from one location and one is from another location, we call this work item a *two-site work item*. For work item 17170, the number of sites is 2: Raleigh and Zurich.

- *Time-zone difference*: A time-zone difference is another indicator of the degree of distribution of a work item. Similar to the number of sites, we use the number of distinct time zones where the commenters are located. Consider as an example a work item which was commented by three different developers located at three different sites. Two are in the same time zone and one is in another time zone. We call this work item a two-time zone work item. In our example, the number of time zones is two because Raleigh and Zurich are in two different time zones.
- *Number of comments*: We interpret the number of comments as an indicator of how large the work item is in terms of effort. The more comments, the larger the work item is. There are four comments in our example.
- *Number of comment authors*: Number of commenters is also an indicator of how large a work item is in terms of effort. The more commenters, the more effort is spent on the work item. In our example, there are three commenters Carl, Joey, and Landner.
- *Number of teams*: Number of teams is also an indicator of how large a work item is in terms of effort. For this measure, we use the number of component teams the commenters are in. Consider as an example two commenters where one is from two different component teams and the other one is from a third component team. We call this a three-team work item. In our example of work item 17170, the number of teams is two: Repository and Work item.
- *Severity*: Severity is an indicator of the work item's urgency. In Jazz, a work item can have one of six values: unclassified, minor, normal, major, critical,

and blocker. The values are encoded by integer values from 1 to 6, where 1 represents unclassified and 6 represents the blocker severity. Severity is set when a work item is created. It can be changed during the life circle of a work item. The severity of work item 17170 is 3.

- *Priority*: Priority is also an indicator of the work item's urgency. The priority of a work item can be unassigned, low, medium, or high. It is represented by integer values ranging from 1 to 4. At Jazz, the priority helps developers when planning and scheduling their work. A work item with a high severity may have a low priority, when its due date is somewhere in the future and not in the current development iteration. The priority of work item 17170 is 4.

For our social network analysis in Chapter 5, we use the work items and the commenters to construct our communication based social network. In the example of work item 17170, we connect Carl, Zoel, and Landner because they were communicating with each other. Figure 4 shows how Carl, Zoel, and Landner are connected within the global communication network because of their communication. More detail on how we construct the communication network will follow in the Social network analysis section below.

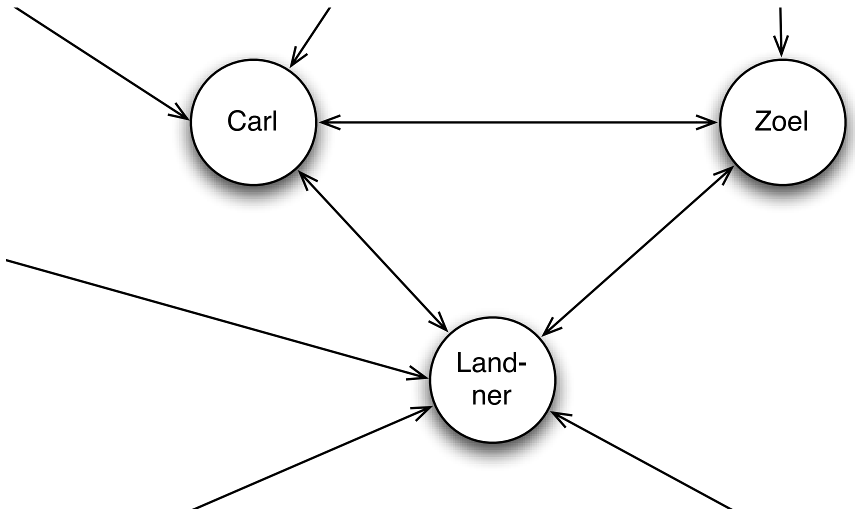

Figure 4: Connections between the commenters of work item 17170 within the communication network due to their communication

Statistical analysis methods

In Chapter 4, we use different statistical analysis techniques on the data constructs, e.g. *resolution time* and *response time*. The choice of statistical tests and procedures depends on the underling distribution of the sample data. To examine the distribution of resolution time or response time, we produced the histograms of the two distributions which are shown in Figure 5 and Figure 6 respectively. For clarity we divided the work items according to their number of sites as described in the Data construct section above. The *n-site* category consists of work items that were commented on by people from *n* number of sites.

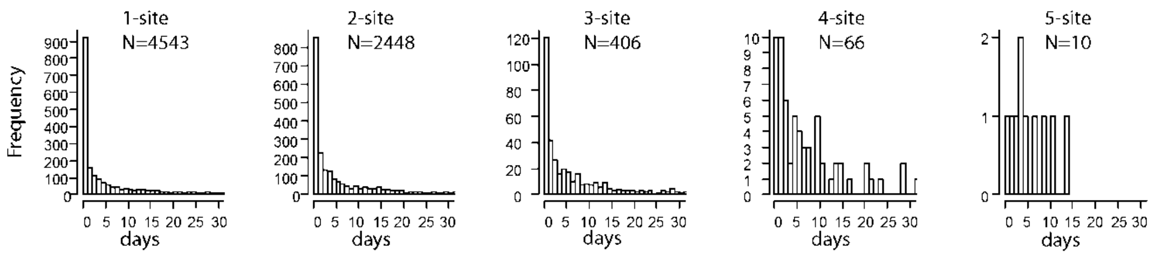

Figure 5: Distribution of *response time* divided according to the *number of sites*.

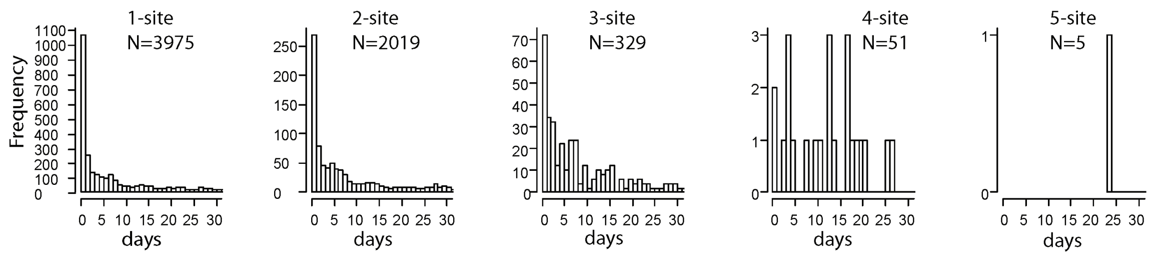

Figure 6: Distribution of *resolution time* divided according to the *number of sites*.

As observed in Figure 5 and Figure 6, the distribution of the two data constructs is not a normal Gaussian distribution. Instead, it is skewed to the left side. We performed Q-Q Plot and a Chi-Square normality test. Both confirm that the distributions are not normally distributed. As a result, in this study, we use non-parametric statistical analysis tests and procedures such as the Spearman's ρ-correlation test, Kendall's τ-correlation test, or the Kruskal-Wallis analysis of variance.

Spearman's ρ-correlation test

The Spearman's ρ-correlation test is the earliest and best known non-parametric test [60]. The test takes two dependent variables of the same objects. For example, if we have three objects with two attributes $x = \{1, 2, 2\}$ and $y = \{5, 2, 4\}$, then the three objects should be (1, 5), (2, 2), (2, 4). Spearman's ρ-test returns a correlation index, index ρ, from -1 to 1. If $\rho = -1$, there is a perfect negative correlation between x and y. This means that for each object, if x is high, y will be low, e.g. (3, 2), (4, 1), (3, 1). If $\rho = 1$, the opposite is true, e.g. (1, 2), (1, 4), (2, 3). If $\rho = 0$, there is no correlation between x and y at all.

Kendall's τ-correlation test

Kendal's Tau is a non-parametric test of correlation or association. It is similar to the Spearman's ρ-test. It returns a τ correlation index. The interpretation is similar to that of the ρ value. If $\tau = 1$, there is a perfect positive correlation. If $\tau = -1$, there is a perfect negative correlation. The Kendall's τ-correlation test is, however, a better test than Spearman's ρ-test when there are many ties and either of the *x* and *y* e.g. (1,3), (1,2), (1,4), (2,1), (4,1).

Kruskal-Wallis analysis of variance

Kruskal-Wallis is another non-parametric test. It is used to check if different sets of data are the same. For example, if we are given four different sets of data points which all belong to a variable, and we want to see if any of the four sets is different from the rest, we can use the Kruskal-Wallis test. In this case, the null hypothesis is that there is no difference among the four sets. The test returns a *p* value from 0 to 1. If the *p* value is smaller than a chosen level of acceptance, e.g. 1%, 5%, or 10%, we can reject the null hypothesis which means that there is a difference among the four sets of data. Otherwise, if *p* is larger than the chosen level of acceptance, we have to accept the null hypothesis that there is no difference among the four datasets.

χ^2-test

The χ^2-test is a test of independence. For example, if we have two data sets *A* and *B* and we want to check if *A* and *B* are similar or different, we can use the χ^2-test. In this case, the null hypothesis of the test is that *A* and *B* are different. If the *p* value is smaller than a chosen level of acceptance, e.g. 1%, 5%, or 10%, we can reject the null hypothesis which means that *A* and *B* are the same. Otherwise, if *p* is the larger chosen level of acceptance, we have to accept the null hypothesis (there is no difference between *A* and *B*).

Social network analysis

As shown in the example in the Data construct section above, we construct a communication based social network using the data from the work item repository. We construct the network using the following rules:

1. Every node corresponds to a team member in the network. Exactly all the members who participated in at least one work item discussion are included.
2. If two members included participated in at least one work-item discussion, we link the corresponding nodes by an edge.

The product is a unique network of team member. In the example of work item 17170, the three commenters are Carl, Zoel, and Landner. We created the three nodes by rule 1. Then we connected the three nodes because they participated in at least one work item discussion by rule 2. This produces the graph shown in Figure 4 which is part of the Jazz team communication network.

In order to gather the information, we wrote queries that collect the data from the work item database to build the network from our database (see the Data collection methods section for more information). The result sets are triplets containing the following: (member a, member b and the number of work items that members a and b discussed). A simple script converted the result into an adjacency matrix which we use to represent our communication network.

Social network analysis in general is a distinct field of study that analyses the pattern of relationship among interactive units of a society [64]. In software engineering, the interactive units are the member of the software team and the society is the team itself.

To explore the communication pattern of the Jazz team in Chapter 5, we will use tests that are related to a specific kind of network called a *core-periphery network*.

To understand what a core-periphery network is, we have to understand what a *sparse network* is. A sparse network is a network of loosely connected components (See in Figure 7 (a)). Note that word component here is defined loosely as parts of the graph that are connected more than other parts. For example, there are three different components of the graph in Figure 7 (a). On the other hand, a core-periphery network has only one strongly connected component. Figure 7 (b) shows an example of the core-periphery network.

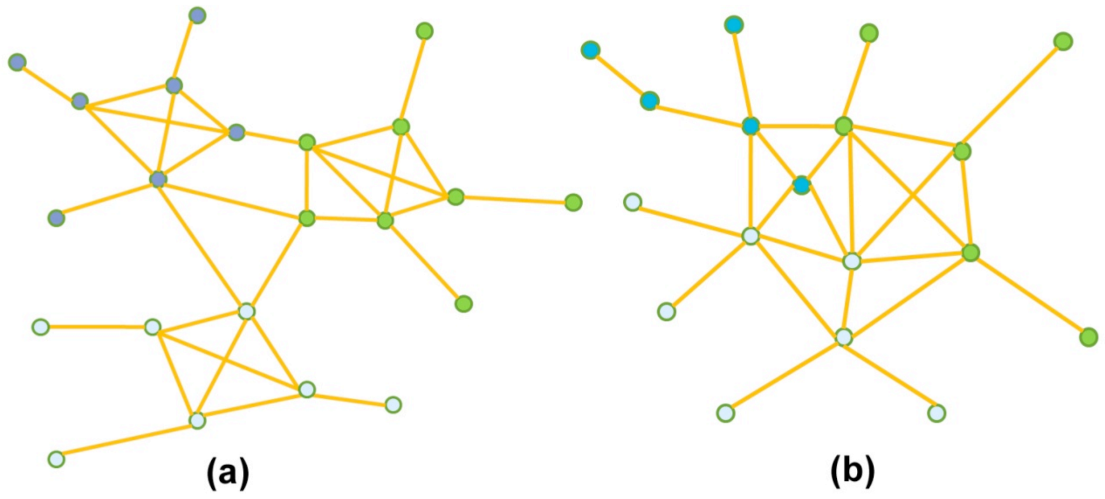

Figure 7: Examples of (a) sparse and (b) core-periphery social network

Core-periphery test

To check whether a network is a sparse network or a core-periphery network, we use a core-periphery test. A core-periphery test determines the degree to how the understudied network represents a core-periphery network. There are many models formalizing the degree of core-periphery of a network [8]. We use the one available on UCINET [3]. UCINET's correlation test returns a number between 0 and 1. A correlation value of 0 means that the network does not resemble a core-periphery network at all. A correlation value of 1 means that the network is a core-periphery network.

K-core

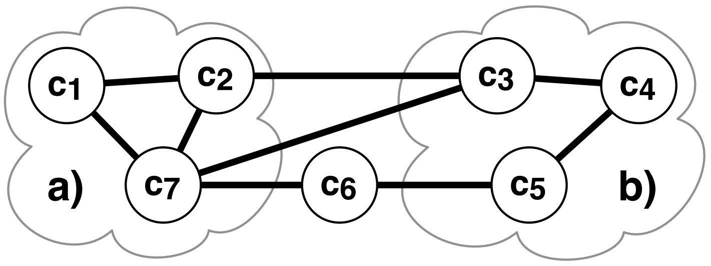

Figure 8: Example how to calculate *k-core*, group degree centrality, and group efficiency

In Chapter 5, we will use the notion of a *k-core*. A *k-core* of a graph *G* is a subgraph of *G* that contains only the nodes having a degree of at least *k*. For example, if *k* is set to 5, each node of the *k*-core must have at least 5 connections to other members of the core. In Figure 8, the *k*-core for $k = 2$ consists of the nodes c_1, c_2, c_3, and c_7, as all have at least two connections to these nodes of the *k*-core subgroup. Note that this example has only one core. Other networks may also consist of multiple *k*-cores. Identifying the *k*-cores with $k = 3$ in Figure 8 will result in an empty group. There is no subgroup in which all nodes have at least three connections to the members of that group.

Group Degree Centrality

The *group degree centrality* index [23] provides a centrality measure for a group of nodes in a network. It is defined as the number of nodes outside the group that are connected to the members of the group. Group degree centrality identifies how central a group of nodes is in relation to the rest of the network.

The group degree centrality is calculated as the number of nodes outside the group that are connected to the members of the group. In our study we compared different groups and as such we normalized the index by the number of actors outside the group. For example, in Figure 8, the group degree centrality of the group a) consisting of the nodes c_1, c_2 and c_7 is 2, as the group is connected to the two nodes c_3 and c_6. The normalized index is 2/4=0.5. Group a) is connected to 50% of the nodes outside the group.

Group Efficiency

The *group efficiency index* [23] for a group provides a measure of how redundant the communication ties from members inside the group to outer members. The group efficiency index is defined as the fraction of the size of the minimum subgroup that has the same group degree centrality index as the whole group and the number of group members. For example, the efficiency for group a) in Figure 8 is 1/3=0.333, as the nodes c_1 and c_2 can be removed from the group without decreasing the group degree centrality index.

Chapter 4: Investigating speed of communication

In this chapter, we present the data analysis and results of our first research goal: the effect of distance on communication speed in the IBM Jazz Team. As mentioned above, delay in communication has been the fear of many software managers when dealing with distributed teams or customers [37]. Herbsleb and Mockus showed that distributed communication can introduce on average a day-and-a-half delay compared to collocated communication [34]. The geographic distance seems to cause communication delay. In this chapter, the research goal is to re-examine whether recent improvements in software engineering practice and tool support have improved this problem. Because the IBM Jazz Team is a software team with experience working in distributed setting and they are using a new collaborative software development IDE, they are prime candidate for our study.

Research questions

To better guide our inquiry on the effects of distance on communication, we ask the following four research questions.

RQ1: Does distribution affect the speed of communication?

As documented in previous research [30, 34, 57], distributed communication is believed to introduce delay as the geographic distance grows. We investigate possible evidence that distribution affects the communication speed in our Jazz data.

RQ2: Do other factors such as the number of comments, the number of authors, the work item's severity and priority of the tasks influence the speed of communication?

Many factors besides the geographical distribution may influence the communication [34]. Within the availability of our dataset, we investigate the relationship between communication and other factors, e.g. the number of comments, number of authors, the work item severity and priority of the tasks.

RQ3: Does speed of communication relate to productivity?

The case studies [30, 34, 57] document that distributed communication introduces delay which in turn decreases productivity. However, the relationship between communication delay and productivity has not (yet) been examined statistically. What is the relationship of the speed of communication and productivity in distributed settings?

RQ4: Does time-zone difference affect the distributed software team?

Follow-the-sun software development is the golden goal for GSD [62]. Because of globalization, company can now open offices in multiple time zones. This brought the hope that developers in different time zones can continue each other jobs because at least one of the sites will be in working hour. However, as we mentioned earlier, research has shown that distribution has a negative effect on distributed development. The Jazz team that we are investigating spans over many time zones. We want to determine whether the time-zone difference has a negative effect on the communication or indeed improved the speed of communication.

Data analysis and result

RQ1: Does distribution affect the speed of communication?

We divided the work items into five different sets based on the number of sites the communication originated from. For example, if a work item was commented by people from Ottawa and Zurich, it belongs to the 2-site category. As mentioned in the Data construct section in Chapter 3, the number of sites is our conceptualization of how distributed a work item is. A 4-site work item is more distributed than the 3-site, 2-site and 1-site ones. The average response time is the conceptualization for the speed of communication. The shorter the response time is, the faster is the communication. With this conceptualization, we compare the average response time of the five different sets of work items to see if distributed communication is slower than collocated communication as documented in Herbsleb and Mocus study [34] which used a similar conceptualization.

Table 4 shows the average response time of work items for different distribution levels. One can see that as the distribution increases, the mean response time decreases. However, the median response time is actually increased with growing distribution. Figure 9 shows the box plot of the data. The middle line of the box plot indicates the median. The box shows the first quartile and the third quartile. The whiskers show the maximum and the minimum non-outliers. Because the distribution is much skewed to the lower end, there are many extreme outliers on the top. We choose not to display this in the graph to avoid cluttering of the figure.

Table 4: Average response time of work item with different distribution level

	N	Mean	Std	Min	25%il	Media	75%il	Max

			Dev		e	n	e	
1-site	4543	18.88	45.82	0.00	0.21	2.43	15.63	672.16
2-site	2448	16.72	41.61	0.00	0.44	3.04	14.36	559.05
3-site	406	13.21	47.81	0.01	0.69	3.70	12.68	899.06
4-site	66	8.42	9.59	0.01	1.58	4.99	10.26	37.38
5-site	10	5.60	4.34	0.82	2.84	4.07	8.10	13.96

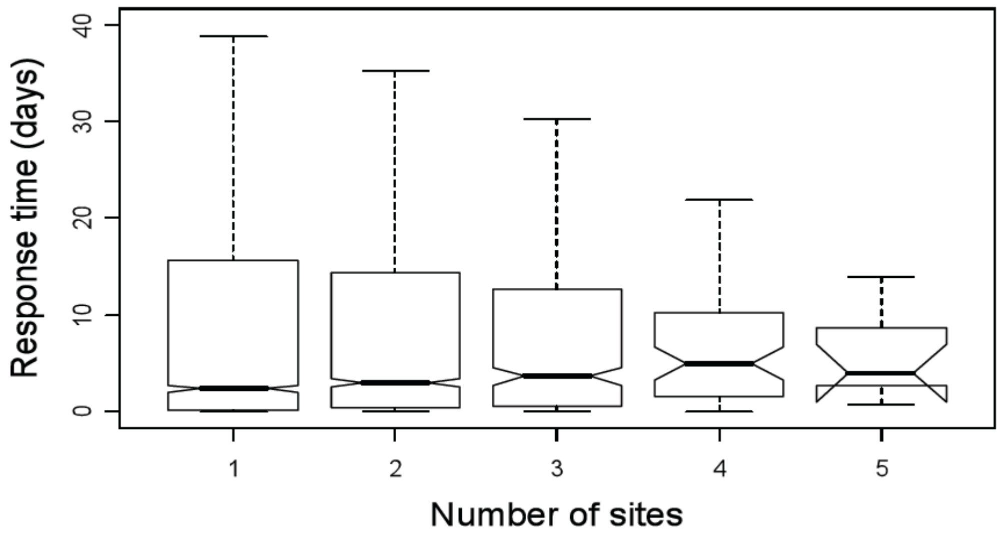

Figure 9: Box plot of average response time

To see the effect of distribution on speed of communication, we perform a Kruskal-Wallis test [58] on the five categories as this test is a non-parametric one-way analysis of variance. The test outputs a coefficient and a *p*-value. As shown in Chapter 3, if the *p*-value is lower than the chosen level of confidence, we reject the null hypothesis implying that there is no difference between the categories. Our test shows that $K = 26.31$ and $p < 0.001$. This means that the five different categories are from different distributions. In other words, this shows that distribution does have some influence on speed of communication.

Knowing that there is an influence, we next investigate the magnitude of the correlation between distribution and speed of communication. To do this, we run Kendall's τ-correlation test [58]. It is a non-parametric test of correlation or association. The test outputs a coefficient τ that ranks from -1 to 1 (as shown in Chapter 3). Our test yields a τ-value of 0.05, implying a very weak correlation.

RQ2: Do other factors such as the number of comments, the number of authors, the work item's severity and priority of the tasks influence the speed of communication?

In order to find out what other factors influence the speed of communication, we ran Kendall's τ-correlation test for all of the work items' attributes. The different attributes were explained in detail in the Data construct section of Chapter 3. The results are shown in Table 5. For example, the correlation between resolution time and number of time zones is 0.5 which is a low correlation.

Table 5: Kendal's τ-correlation of different factors

	1	2	3	4	5	6	7	8	9
1.Response Time	1.00	0.70	0.07	0.10	0.03	0.06	-0.13	-0.05	0.00
2.Resolution Time		1.00	0.20	0.14	0.05	0.11	-0.16	-0.11	0.01
3.No. of Comments			1.00	0.65	0.26	0.40	0.14	0.04	0.10
4.No. of Authors				1.00	0.39	0.57	0.11	0.02	0.09
5.No. of Time Zone					1.00	0.74	0.09	0.04	0.02
6.No. of Site						1.00	0.11	0.02	0.11
7.Severity							1.00	0.15	-0.02
8.Priority								1.00	-0.07
9.Category									1.00

RQ3: Does speed of communication relate to productivity?

Since proving a causal relationship is difficult, finding the causal effect of communication speed on productivity is not easy. However, we can investigate the correlation between the two factors. As in RQ1, we conceptualize the speed of communication by the work items' average response time and productivity by the work items' resolution time. We run Kendal's τ-test [58] on the two factors. If the test returns a strong correlation, then speed of communication is linked with productivity. The test returns indeed a strong correlation with $\tau = 0.70$.

RQ4: Does time-zone difference affect the distributed software team?

We conceptualize the time zone difference by the number of time zones for each work item. As defined in Chapter 3, this number is the number of time zones from which the developers who commented on work items come from. If the time-zone difference helps improving the communication speed, we expect a negative correlation between the work items' number of time zones and the work items' average response time. Otherwise, we expect a positive correlation. Figure 10 shows the average response time for work items from different numbers of time zones. We can see that the response time slightly increases with the number of time zones, while the variation decreases. Kruskal-Wallis shows that there is a difference between the four categories. It returns $p = 0.022$. However, Kendall's τ-correlation test returns an extremely low correlation of $\tau = 0.02$. Therefore we can neither say that the time zone affects the response time nor that it improves communication.

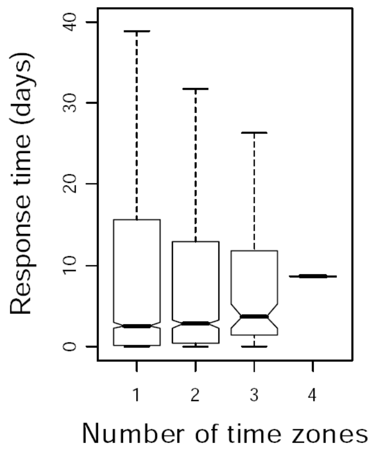

Figure 10: Average response time of work items from different time zones

Discussion

In regard to RQ1, we found that the geographical distribution did not affect the communication speed as much as documented in the literature. The numbers in Table 4 show that the mean response time actually decreases with the number of sites involved in the work items. The median, on the other hand, increases with growing distribution. This is conflicting information because the distribution of the response time is extremely skewed to the lower end. Figure 11 shows the histogram of the response time distribution. Note that most of the work items are finished within a day.

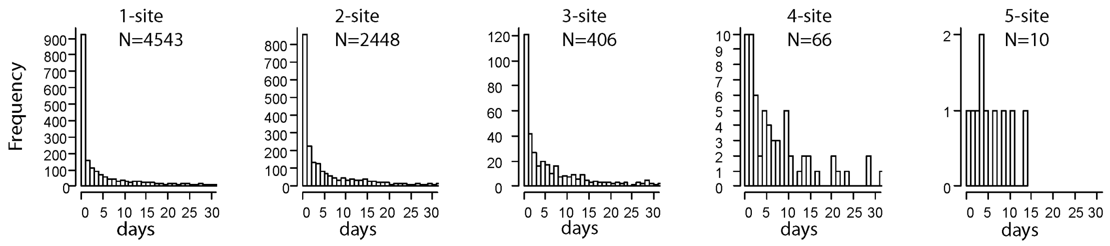

Figure 11: Histogram of response time

Our statistical tests confirmed this. The Kruskall Wallis test shows that the five different categories are different for $\rho < 0.001$. This indicates that, statistically, there are differences in the response time of work items. However, Kendal's τ-test shows that correlation is very weak at 0.05.

This came as a surprise to us. Both practitioners and researchers have reported that delay in cross-site communication has been a big challenge in GSD. We think that the result from our study was different because:

- Other studies [34, 38] use self-report surveys. The responses were mostly qualitative. They might be subjected to memory bias. Our study uses quantitative data from the work item repository. This limits the biases.
- The evolution of computer supported collaborative software engineering helps eliminating the collaboration problems with distributed teams. Jazz is one of the newest software-development collaboration tools. It aims to bring the collaboration aspect into the Eclipse IDE. The Jazz team develops and uses the tool. Thus the tool was specifically designed to work with their distributed setting. This helps improve support for distributed development process.

RQ2 did not reveal more insights other than those that we have already expected. For example, work item's severity has a mild negative correlation with response time (τ=-0.13) and resolution time (τ=-0.16). This makes sense because for higher severity work item, we would expect a lot of fast responses to resolve work item faster. This also reflects on the positive correlation with the number of comments (τ=0.14).

For RQ3, we confirmed that speed of communication affects productivity. The Kendall's τ-correlation test returns a strong correlation with τ = 0.70*.* This means that there is a strong correlation between speed of communication and productivity. This is, however, not a causal relationship. We cannot conclude that speed of communication affects productivity or vice versa. We can say that if communication is slow, productivity will suffer. Or we can say the larger the work item, the longer it takes people to communicate on it.

As for RQ4, we found that the time-zone differences do not strongly affect the response time. The Kruskal-Wallis test shows that there is no significant difference in the response times for work items coming from different time zones. Kendall's τ-test shows a very small correlation of 0.02 between response time and number of time zones. If distributed development is beneficial, we would see a difference in the result of the Kruskal-Wallis test and a strong negative correlation in Kendall's τ-test which means that GSD is faster in responding to request. Our study shows that distributed communication is just as fast as collocated communication.

To separate the effect of distribution conceptualized by the number of sites and the effect of time-zone difference on response time, we performed another analysis. We divided the work items into two categories: those that are commented by developers in one time zone and those that are commented by developers from different time zones. We then examined the effect of number of sites and response time for each category using Kruskal-Wallis. We found that for the work items from only one time zone, distribution has a small positive correlation with response time. Kruskal-Wallis returns a *p*-value < 0.001 which means that there is difference. Spearman's ρ-test returns a low correlation of 0.058. For work items that are commented on by developers from many time zones, the Kruskal-Wallis test returns a *p*-value of 0.19 > 0.05. This means that distribution did not have a statistically significant effect on response time when the developers are in different time zones.

Chapter 5: Investigating Social Network based on Communication

The results from the previous chapter, Chapter 4, show that communication within the IBM Jazz Team is highly successful in overcoming the difficulties of distance and time-zone differences. Our next step is to find out why. To achieve this, we decided to investigate the structure of the communication network. This is our second research goal.

As mentioned in the background, when two team members exchange conversations regularly, regardless of the medium (for example: face-to-face, online chat, or work items' comments), have established a better working relationship to each other than to those in the team they are not communicating with. Through communication they become aware of each other's activities, exchange technical expertise, influence each other decisions, and ultimately coordinate their activities. As the result, social network analysis has been long instrumental in discovering coordination patterns in organizational studies (e.g. [31, 29]). In software engineering, social network analysis has been used to study software teams in particular. Examples include the study by Gloor et al. [28] that reveal different network characteristics of W3C working groups. In distributed software development teams in particular, the study of social networks proved useful in understanding patterns of collaboration and coordination in global teams [63, 38]. In these articles, different measurements of the distributed development teams' social network such as density, network centralization or social technical congruence, were used to characterize the teams' performance such as efficiency of communication or easy coordination.

Following a similar methodology as those studies, we built the social network based on communication of the Jazz team using the method described in the Social network analysis section of Chapter 3. We call this network the *communication based social network* or in short *communication network*.

Research questions

RQ 1: Is the communication network of the IBM Jazz Team a core-periphery or a sparse network?

In a study [38] of 33 distributed and co-located research and development teams in multinational company, Hinds and McGrath found that such team perform better in term of ease of coordination. The web based survey showed that “a more hierarchical structure of *receives information* was associated with more coordination ease in distributed teams”. The core-periphery structure is a close conceptualization of a hierarchical structure. As discussed in the Social network analysis section of Chapter 3, a core-periphery network resembles a star network which is an absolute hierarchical structure. This implies that there is a formal or informal hierarchical structure of communication. We investigate if the IBM Jazz Team is a core-periphery or a sparse network. If it is a core-periphery network, this will explain why the team was able to efficiently manage their distributed coordination.

RQ 2: How connected are the members in a geographical location to other project members in the IBM Jazz Team?

Collocated team members normally work closely together compared to their relationship with other members who are distributed over the distance. On the other hand, distributed members may never meet each other face-to-face. This distance barrier can potentially create problems such as loss of "communication richness" [6], misunderstandings [53], or delay in communication [34]. Using the communication based social network we constructed, we want to find out how each collocated team is connected to the rest of the Jazz team.

Data analysis and results

RQ 1: Is the communication network of the IBM Jazz Team a core-periphery or a sparse network?

To find out if the IBM Jazz communication network has a hierarchical structure, we use two methods: using visual inspection of the network using a k-core annotation and the use of core-periphery test. Both of the methods were explained in Chapter 3.

Figure 12 is the visualization of the communication network of the IBM Jazz team. Each node is a Jazz team participant. If a participant has commented on the same work item with another participant, there is a link between them. The color of the node indicates how connected the participant is to the rest based on their *k-core* membership (see Social network analysis section in Chapter 3). We also circle the participants who work at same location among seven main development sites. Note that there are participants that are from other locations as well. For demonstration, we highlighted the *k-core* with $k > 25$ is highlighted with a red circle in the center to indicate the highly connected nodes. The core has a membership of 60 out of 112 members in the entire Jazz project, and a number of 2118 ties out of a total of 3296 ties in the network. This indicates a large core including almost half of the members and about 75% of the communication in the project. Visually, this is a core-periphery network.

Secondly, we run the core-periphery test (see Social network analysis section in Chapter 3) on UCINET. This correlation test returns a number between 0 and 1. A correlation value of 0 means that the network does not resemble a core-periphery network at all. A correlation value of 1 means that the network is a core-periphery network. The test on the IBM Jazz team communication network yields a correlation of 0.758. This means that this network is a core-periphery network.

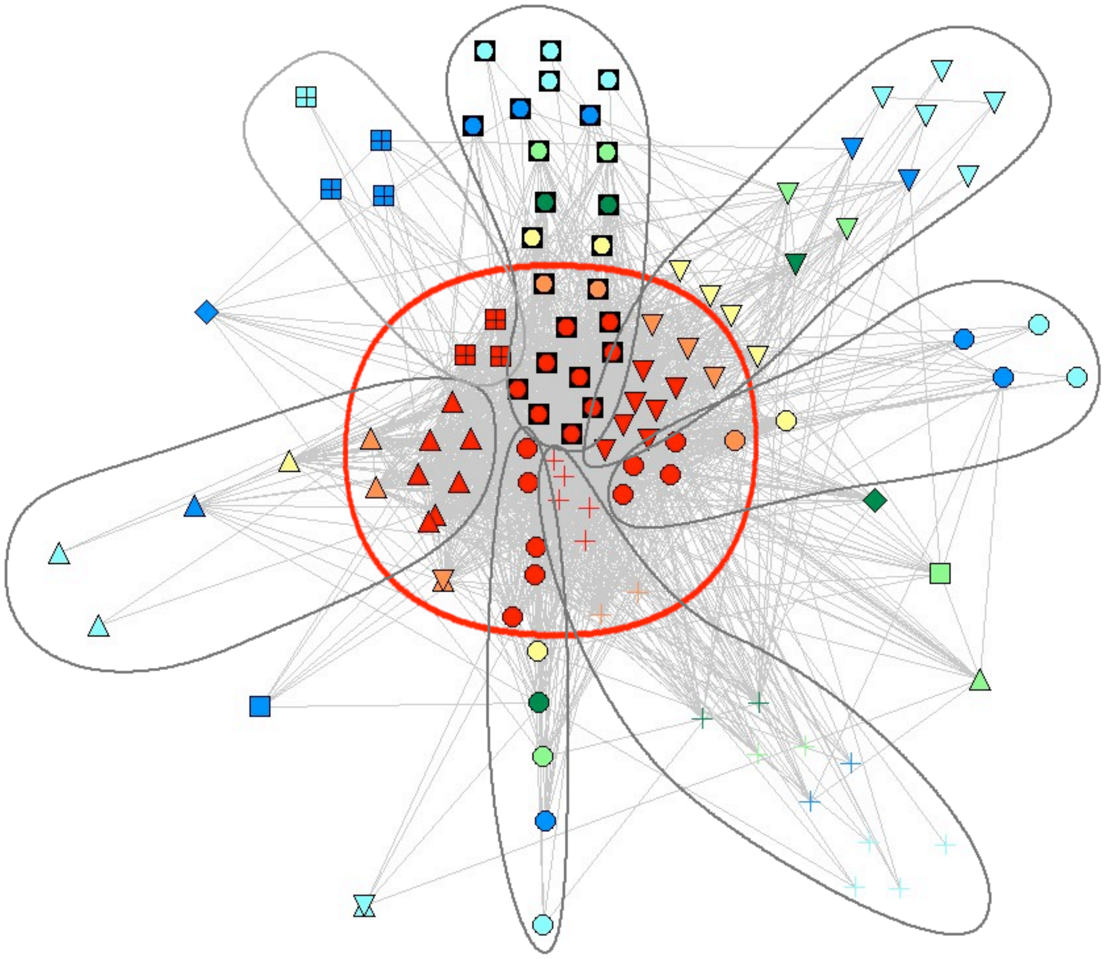

Figure 12: Visualization of the IBM Jazz team communication network

RQ 2: How connected are the members in a geographical location to other project members in the IBM Jazz Team?

To examine the level of connectedness of each geographical location to other project members and to examine how central each geographical location is compared to other locations, we computed for contributors in the network of each geographical location the group degree centrality and the group efficiency index respectively, as follows.

Table 6: Group Degree Centrality index and group efficiency index of the IBM Jazz component teams

Group Location	Number of team members	Group degree centrality	Centrality efficiency
Zurich	12	97%	0.25
Raleigh	17	97%	0.41
Lexington	23	94%	0.26
Ottawa	25	93%	0.20
Beaverton	10	83%	0.40
Toronto	10	75%	0.50
Hawthorne	7	55%	0.57
Average		85%	0.37

As discussed in Chapter 3, the group degree centrality index provides a centrality measure for a group of nodes in a network. It is defined as the number of nodes outside the group that are connected to the members of the group. Group Degree Centrality identifies how central a group of nodes is in relation to the rest of the network. For example, if Site A has a group degree centrality of 82% and Site B has a group degree centrality of 45%, then Site A is appeared to be more important in managing the communication across the entire team. To compare the geographically different groups in Jazz we normalized the index by the number of actors outside the group and thus report percentages as values of this index. More details on this measure and an example is provided in Chapter 3. The Group Degree Centrality index for each of the seven Jazz development locations are shown in Table 6. The table also shows the number of participants at each location. For example, 12 project participants are located in Zurich and have communicated to 97% of all project members that are at a different location than Zurich. As almost all locations have a very high group degree centrality index and the indices of the top four locations (>90) are almost equal, we cannot identify geographical locations that are more central than the other locations in terms of number of external communication peers.

The group efficiency index provides a measure on how redundant the communication ties are from members inside the group to members outside the respective group. It is defined as the fraction of the size of the minimum subgroup, which has the same group degree centrality index as the whole group, and the number of group members. More details on this measure and an example is provided in Chapter 3. This index ranges from 0 to 1, where 1 indicates that the group has a lot of redundant communication ties to project members outside the group and 0 indicates that the communication in the group is very efficient. In Jazz, the efficiencies of the seven geographical locations range from 0.2 to 0.57 (see Table 6) and have an average of 0.37. The overall low efficiency index indicates efficient communication ties across project participants from different geographical locations.

Discussion

As for RQ1, we found that the project-wide communication-based social network has a large core of active contributors. This suggested that many project members were actively communicating across functional teams as well as distances. Since maintaining interpersonal relationships and awareness of work and expertise have been found challenging in distributed teams (e.g. [22, 30, 33]), a less dense network should be expected in such a large distributed team. Distributed social networks are typically significantly smaller than same-site networks, with a restricted flow of information across sites [22, 34]. The presence of particular team members acting as information broker through whom much of the team communication flows to the distanced teams, was found to ensure the efficiency of coordination across teams [38].

We observe that the Jazz network, in contrast, is dense across distances and functional teams. While we did not compare the properties of same-site vs. distributed networks in Jazz, we find that the distributed network has a large core. Instead of having multiple clusters of connected cores, each associated with a geographical location in the project, the entire project communicates through one core. With a core comprising about half of the entire project team, the network in Figure 12 also shows that about half of the members from each geographical location are in the core. Having multiple members of each team in the core, connecting their team to others, reduces possible communication bottleneck and introduces redundant communication channels, enabling fast communication. While these core members may act as information brokers to the rest of the team, the network as a whole exhibits a rather informal hierarchical organization since the other peripheral members in each geographical location appear to be fairly connected to the other members that are in and out of the core area.

Our conjecture is that this particular project-wide communication structure was enabled and continually supported by the above-mentioned communication practices, substantially contributing to reducing the communication delay causing by distance.

For RQ2, we found that all of the seven main development sites maintain a similar high degree of connectedness to other sites in the IBM Jazz team. This is demonstrated by the higher than average group degree centrality index across the seven sites (55%-97%) (see Table 6). The average group degree centrality index for the seven sites is 85%. This means that, as opposed to communicate within its own team, each of the sites was able to maintain adequate communication to the other distributed sites. This contributes to our conjecture in RQ1 that the IBM Jazz Team was able to establish a project-wide communication structure. This explains why distance does not have much affect on communication in the IBM Jazz Team as found in Chapter 4.

We also found the communication efficiency as defined by the group efficiency index is high (<0.50) for five out of seven sites (see Table 6). The average overall group efficiency index is 0.37. This index measures of how redundant are the communication ties from members inside the group to members outside the respective group. The low index numbers indicate that each of the site member hold important communication role in that team. In practical terms, it is efficient because each of the team members acts as an important broker of information in the network. If the index is high, it means that the member can communicate with other member of the same team the same question. Though this could indicate redundancy in communication, it is inefficient because, to conduct communication effectively, each member has to spend time familiarize with each other work.

Chapter 6: Discussion

Discussion

In this thesis, we describe the results of our case study on the communication of large globally distributed team: IBM Jazz Development team. Our goals were to study a) the effects of distance on communication speed and b) the structure of the communication network. While the results of each research goal were discussed in Chapter 4 and Chapter 5, we want to emphasis in this chapter our main findings and the indications of our results for practitioners.

Distance may not matter for the IBM Jazz Development team

In contrast to previous study [34], we did not find that geographical distance introduces significant delays in communication. Based on the previous literature, we expected that response times would be longer as the number of sites involved in the communication increased. Thus we analyzed the response and resolution times in relation to the number of sites involved in each work item. Our findings show no clear impact of distance. Although the test of difference, namely the Kruskal-Wallis test, showed a statistical significance (indicating that the variation in **one** of the distributed communication settings is different from the others), the correlation results that tested the size of this effect were extremely low.

The descriptive statistics in our data set also allows us to do a more in-depth analysis of the communication trends in same-site and distributed communication. The distribution of response times for the five different categories of work items is shown in Figure 11. With the exception of the 5-site category, the 1,2,3 and 4-site categories show a consistent pattern: about one third of work items have an average response time of one day; about 10% of work items have an average response time of 2 days (exception is in the 4-site category), and the remaining work items exhibit a long-tail distribution pattern indicating the presence of some comments with very long response times. This distribution, together with information shown in the box plot may explain the inverse trend in the pattern of mean vs. median shown in Table 4: the number of long responses decreases as the number of involved sites increases (mean value decreases as number of sites increases), whereas there are fewer quick responses as the number of sites increases (median value increases as number of sites increases).

Further, as our data are not normally distributed, the median is an important measure for the trend in response time (as opposed to the means). As such, one can observe that the median response time in the 2-site distributed communication category is only 0.61 days larger than the median response time in the same-site communication category. We believe the median indicates that collaborators in the 2-site communication category wait on average for a response about half a day longer than those in the collocated communication category. When this is considered in light of a 6-week long iteration cycles, one may understandably not find it practically significant. A similar trend can be observed as the number of sites increases. The 5-site case is somewhat difficult to analyze. With a small sample size of 10 work items, the response time distribution shows an almost even distribution of the response times from 1 to 13 days. However, our sample available is too small to allow us to regard this as representative.

In summary, we believe that the geographical distribution did not have a practical effect on the response times in the Jazz environment.

One possible explanation for this difference in our findings is that previous studies [34, 38] used qualitative, self-reported interview or survey data, which might not have provided an objective measure of communication delay. The interviewed participants might have remembered and reported only about communication instances with a strong delay, possibly a small and unrepresentative sample to generalize from. Our data are less prone to participant recollection problems.

Another possible explanation might come from the characteristics of the Jazz project participants. We can expect that developers developing a tool focusing on collaboration support reflect on how they communicate and collaborate within the project and optimize these activities. Thus, they may perform better than developers in other projects, who focus on a different domain and deal with distribution issues as lower priority items. In addition, demographic information such as age and familiarity with text based communication specific to commenting on work items or chat might influence the ability to collaborate across sites. Unfortunately, we do not have the demographic information of the Jazz team.

Another possible explanation is that tools (such as Jazz) that aim at integrating project management, communication and coding were able to bridge the gap of distribution. Jazz aim is to provide greater traceability between the many artefacts in the development environment and thus assisting coordination and communication. Coordination and communication is not just an explicit act. For example, when two drivers stop at a 4-way stop intersection, they did not explicitly coordinate and communicate their activities. The communication and coordination were carried out implicitly via a set of predefined traffic rules and the signalling device, the stop sign. In the case of global software developments, tools such as Jazz provide the signalling devices aiding communication and coordination between distributed team members. The devices come in form of software artefacts such as work items and builds. Consider the example when two developers have to work on two interdependent work items. Instead of having to tell each other when and how the dependee work items will be resolved, the developer who is in charge of the dependent work item can just follow the progress of the dependee work item on his work space. Together with a good the set of rules, in form of a well-defined process, the Jazz team was able to successfully carry out their communication and coordination leading to the success of their product.

Communication structure of well functioning distributed software engineering teams

In studies of distributed software development teams, social networks have been used to understand patterns of collaboration and coordination in global teams. Representations such as social networks allow software engineering research to capture information about the real world relationships that form among people [19]. For example, Cataldo and colleagues [13, 12] used of social networks based on actual communication and task dependency to calculate social technical congruence measure. This measure can be used to indicate the degree to which the actual social structure of the software team matched the coordination requirements of the software they built. Marczak et al. [50] make use of social networks in the context of requirements engineering to study the information flow among software developers and identify key brokers of information among the developers.

In our study, we also use social network analysis to understand the communication structure of the IBM Jazz Team. We constructed the communication based social network using the communication data from the team's work item repository. This has an advantage over using interviews and surveys because it is not subjective with respect to the interviewee memory bias. As explained in the previous section, the interviewed participants might have remembered and reported only about problems in communication, possibly a small and unrepresentative sample to generalize from. Using a communication network constructed from a repository, we can measure more objectively how the communication was carried out by the team, in particular, obtain quantitative measurements on the team's communication structure. It is very time consuming to survey or interview each of the developers in the Jazz team for how many times or how many people they have been in communication with. It also allows us to cover a larger number of subjects in a short amount of time. This is important because a distributed development team normally involves a lot of people in many teams at several different locations. It was impossible for us to construct a communication network of this particular team using interviews or surveys in the two years time frame of this study. However, our approach does have its disadvantage. We found that it is hard to interpret our research results without either formal or informal confirmation by the team members.

As explained above, we found that the team was successful in eliminating the effect of distance on their communication speed. Because of this, we conjecture that the communication structure we found in the IBM Jazz Team is ideal for a distributed software development team. First of all, there should be a large core of active contributors that contains members from across different geographical locations in the project-wide communication network. In the Jazz Team, we found a core comprising about half of the entire project team (Figure 12). Half of the members from each geographical location are in the core. The existence of this core is important because the core members can potentially maintain a high awareness of the states of the project due to their active roles in the communication network. As a result they can act as information brokers to other members at their sites. Second of all, each of the geographic location should exhibit a strong connection with other locations. We found that each of the seven Jazz sites has a higher-than-average group degree centrality index (55%-97%, see Table 6). This means that each of the sites was able to maintain adequate communication to the other distributed sites. This is an important indicator of a good communication structure because strong communication links between teams are important to maintain awareness of project-wide issues and help the members to coordinate their tasks, or resolve cross-site dependencies. Lastly, each of the contributors should have a specialized function in the communication network. In the Jazz Team, we found that the communication efficiency, measured by the group efficiency index, is high (<0.50) for five out of seven sites (see Table 6). This means that each member of a site holds an important role in the communication network. If he or she is removed from the network, the information that he or she could provide to other members is lost. Although it makes sense to have some redundancy in a network, redundancy also can imply inefficiency. For two team members of the same site, to play the same role in the communication network, both would have to spend time to familiarize with each other's area. This time could potentially be used for more productive tasks.

Implications for practice

As in other organizations, communication is a very important process in globally distributed software development teams. Previous studies found that globally distributed development teams in the past have suffered from many communication challenges such as the loss of "communication richness" due to geographic distance [6], misunderstandings caused by cultural differences [53], or delay in communication [34]. In this study, we conducted a relatively new distributed development team: the IBM Jazz Development Team. The team adopted strategies, learned from previous experiences, to alleviate the problems associate with distributed development. The team further implemented new development tools that facilitate coordination and communication in distributed settings. With the positive results shown and discussed in the previous chapters, this case study indicates:

- Practitioners can use communication speed as an indicator of their team's productivity.
- Software teams should be able to overcome the difficulties of distributed development as long as they have the right tools and processes to accommodate the distance factor.

In Chapter 4, we looked at the communication delay of the IBM Jazz Team. An important insight is (RQ3) that we found that the communication delay, measured by the work item's comment response time, has a significantly strong relationship with productivity, measured by the work item completion time. Because it is a correlation, this relation can be interpreted in two ways: (a) if a distributed team's communication delay is low, their productivity is high or (b) if the software team productivity is high, their communication's delay should be low. Either way, this indicates that practitioners can use communication delay as an indicator for their team productivity. If (a) is the case, then the communication speed influences the team's productivity. The manager can then monitor the response time of the team and make sure that it stays inside of an acceptable range. When the communication delay is low, productivity is high. If (b) is the case, then managers can use communication delay as an indicator for task resolution problem in the team. Communication speed can be detected easier than team's productivity. It can be measured by collecting the delta time between memos, emails, or software artefacts' comments as in Jazz. The managers can then derive a metric to detect a dangerously high communication delay in different areas of their distributed team. This gives them a way to prevent more serious issues caused by communication problems that might have been affecting his or her team.

In Chapter 4 and Chapter 5, we also found that distributed development does not affect communication as much as the literature has suggested in the past. This means that practitioners should be able to alleviate of problems of distributed development setting with the right processes and tools. In RQ1 of Chapter 4, we found that distribution has a very low effect on communication speed. With Kendall's τ-correlation of $\tau = 0.05$, the effect of distance on response time is negligible for practitioner to be afraid of distributed development. This is an improvement from what has been reported earlier when company began to enter distributed software development. Chapter 5 showed that the social network of the IBM Jazz team posed similar characteristic of core-periphery network which is known to be better for coordination than a sparse network. These results support our claim. Because globally distributed development became a necessity for large software companies, this is good news for software practitioners who want to take advantage of the benefit brought by distributed software development such as reducing production cost, access to high-skill labour market, and reducing the distance to the customer.

Manager should make sure that proper communication practice and tool support is implemented. As discussed in Chapter 2, many researchers have published recommendations on strategies to avoid problems in distributed development. For example, Carmel and Agarwal [10] suggested tactics to alleviate the impact of distance: reduce the number of tasks that require intensive collaboration, reduce culture distance by enforcing a common organization cultural, and reduce temporal distance by utilising asynchronous communication channels. Lings et al. [49] also recommended strategies for successful development. Although this is not part of our formal study, we have observed many of these strategies implemented by the Jazz team. For example, a Jazz team lead told us that they tried to make sure that developers who work in the same component are also collocated. This is the first tactic suggested by Carmel and Agarwall [10]. Another example is Lings et al. [49] sixth strategy namely to manage processes. This strategy suggests that the distributed team should identify a project leader with full responsibility. This leader should support the local project managers. It is also suggested that regular teleconferences and regular developer reports should be used to monitoring project status. The different teams should try to conduct plan meetings during overlapping working hours. Based on our observation and communication with them, we can see that the Jazz team has adopted this strategy fully. Same goes for tool support. Another strategy suggested by Lings et al. [49] is that a common software configuration management should be used across distributed sites. Comment fields should be used as extra form of asynchronous communication. We observed that this strategy was particularly adopted by the IBM Jazz Development team. Every site uses the same communication tool: the IBM Jazz platform. Each of the artefacts in Jazz contains fields for developer to add comments to the artefact. There are other commercial [45, 41] and open source / academic tools [16, 43] that can support this. Thus, software managers should carefully consider when their company decided to introduce distributed development to alleviate the disadvantages of distribution.

Threats to validity

Despite our efforts to collect data and choose the good analysis in this thesis, our method suffers from several threats of validity. In this section, we describe the threats to allow the reader to make a better judgement about the application of the results and the study itself.

The first threat to validity is the ignoring of others communication channel such as face-to-face conversations or telephone calls from the data set. The IBM Jazz development team has three different online communication channels: the work item repository, the development mailing list, and the chat tools such as IRC or IBM SameTime. We were not able to access the chat records of IRC or Same Time. We further did not include the communication on the mailing list because at the time of the data collection, the mailing list had been restricted for announcements related to team wide integration builds. Aside from online communication, the teams also used face-to-face communication, telephone and voice and video conferences to communicate with each other. Our informal observation when we worked briefly with the Ottawa team confirmed this. Unfortunately, we could not access any of these other communication channels. To minimize the threat, we discussed this with a team lead informally at the beginning of our study. He indicated that the team uses the work item repository as the main communication channel and minimizes discussion outside the work item repository. Because they built the tool themselves, they want to make sure that it is used as it was supposed to.

Another threat to the validity of this study is the absence of certain communication data from the work item repository. Because the team self-hosted the tool they built, we found that some information including work items were changed or deleted during subsequent migration to the new version of the tool. This happened especially at the beginning of the development. We found that the majority of the work items we had to remove due to missing member-information are from the early development (see the Data collection methods section in Chapter 3 for more information). Unfortunately, we could not find a countermeasure for this threat. However, the amount of valid work items are 13,742 of the total 18,618 which is 74%. This is a majority of the work items. Also, most of the invalid work items were in the earlier stage of the development. Therefore this threat is not applicable to the work items in the later development phase.

In a recent published paper, Aranda and Venolia [4] followed up on surveys and discovered that there are many more people involved in a bug tracking system, which is very similar to that used in Jazz, than those who actually appeared in the system. This means that there might be more people involved on a particular work item than those who commented in the work item. So the threat is that maybe the distributed unrecorded communication is delayed much higher than those recorded. Also, when taking into account the possible unrecorded communication, the communication structure might be very different from the one we found. Unfortunately, we could not find a way to counter this threat. However, as mentioned above, the team said they tried to use the work item repository as much as they can because they are testing at the same time.

Chapter 7: Conclusion

In this thesis, we investigated communication in global teams in times when advances in development environments that support the collaboration of global teams are making into to the software industry. We described our case study on communication at the IBM Jazz Development team. The Jazz team is a large globally distributed team with 151 members in 16 different sites in North America and Europe. Being distributed, they face many challenges as documented in the literature: breakdowns in coordination [18, 6], the lack of a common (formal or informal) communication channel [17, 47] or miss-matches between the required and actual coordination [13]. The development team is, however, a relatively new team. The team members are all very experienced in distributed software engineering. The team had implemented many of the newly learned strategies suggested in the literature such as reducing the number of tasks that require intensive collaboration or having a top level manager for all sites [49, 10] to alleviate the problem of distribution. They further implemented software development tools i.e. the Jazz platform [41], providing support for distributed development. With the new practice and tool support, our thesis's goal was to investigate whether distance really matters any longer?

To address this question we derived two particular research goals. The first research goal is to determine the effect of distance on communication speed. Delay in communication has been the fear of many software managers when dealing with distributed teams or customers. Studies in the past showed that distributed communication can introduce on average a day and a half delay compared to collocated communication. We want to find out if the Jazz team still suffer from the same effect of distance as reported. The second research goal is to determine the underlining communication network structure of the Jazz team. Study [38] found that certain structure of the communication network facilitate an improved coordination in globally distributed teams. With the new processes and tools available for their team, maybe the Jazz team was able to achieve the proper communication structure for distributed team communication. We want to find out what the structure is suitable for distributed development. On the first research goal, our study surprisingly found that, in contrast to previous studies, geographical distance does not introduce a significant delays in communication. On the second research goal, we found that the project-wide communication-based social network has a large core of active contributors.

Contributions

When we start this study, the overall aim is to study and understand communication in globally distributed team. As we narrowed down our research into two small goals, we maintained the main purpose of we start with: contribute to the existing knowledge on GSD research. We believe that the insight we got from this study will benefit both the research communication and the software practitioner.

To the research community, this study expand the existing knowledge about communication in distributed software engineering teams.

Firstly, the effect of the distance on speed of communication had been reported to be a problem for distributed team [34]. Our study showed that this effect can be minimized as done in the Jazz Team. The Jazz team in general does not suffer from the effect of distance in distributed development. Although this is only a case study, it challenges the assumption in literature that communication is always difficult for distributed development team. We conjectured that the improvement in processes and tools support indeed help alleviate the effect of distant in the project. We also think that the experience of the Jazz team and the practices that were implemented by the manager helps the team to achieve such efficiency in their communication. Tools support is another possible explanation. The Jazz development platform was specifically designed to provide communication and coordination support for distributed software development. We think that the tool successfully enable the team communicate effectively even though they are distributed over seven different sites.

Secondly, we show that the communication network in our case study has a hierarchical structure. Although the communication was carried out organically without such a defined structure, our analyses revealed the Jazz team communication structure is a core-periphery network with a large core with high group degree centrality and group efficiency for each site. We believe that this project-wide communication structure was substantially contributing to reducing the communication delay causing by distance. We conjecture that this is the proper structure for a globally distributed development team.

For the practitioners, e.g. software managers or team leaders, our study provide insights into the way distributed team works. Because we found that communication speed are not significantly affected by distribution, we believe that practitioner should be able to establish a successful distributed development environment as long as they implement the right process and have the proper tool support. We also conjecture that practitioners should try to monitor communication speed as an indicator of their distributed team's productivity by establishing metrics based on communication speed. The metric can be designed to detect dangerously high communication delay in different areas of their distributed team. The manager and other team members can then use this as an indicator of communication problems and prevent more serious issues that might arise due to the occurring problems.

Future work

Due to the time limit, the scope of this thesis cannot cover all of the topics it set out explore: communication in software teams. There are much left to follow up: confirming the results with other case studies and provide further explanations of the result.

This thesis is a case study of communication in a large software engineering team using a particular collaboration tool. The claims here are valid to all large software engineering teams. So other case studies are much need to confirm the results. The appropriate case study to confirm the results here should target a large software engineering company that has at least three or more locations distributed globally. The company should use a single work items tracking tool. This tool should be used in the company as the primary communication channel between team members regardless of the team or the location of the members. An industrial-wide survey is also needed to confirm the results especially about the effect of distance on communication.

There are a few results claimed here should be investigated further. First, we found that distance does not matter when the right tools and processes are used. The questions here are: What tool is right for distributed developments? Which process is right for distributed developments? To answer the first question, a comparison study should be done on similar distributed software team using different communication tools. The second question is rather difficult to answer by a comparison study. While the evaluation of tools is already hard to study, the evaluation of process is much harder. The outcome of different processes are hard to judge because there are so many confound factors. The answer though may eventually emerge as practitioners try to adapt the development processes for distributed development. We also found that there is a strong relationship between the speed of communication and task completion. As communication is much easier to track than performance, there is a potential to use communication related metric such as comment frequency or different properties of the communication based social network to aid software project management. But first, follow up studies have to confirm and explain this relationship.

Bibliography

[1] *MySQL:: The world's most popular open source database*. 1995-2008.

[2] *Offshore's New Horizons*, in *Global Technology Business*. 2000. p. 12-15.

[3] Analytic Technologies, *UCINET 6: Social Network Analysis Software*. 2007: Lexington, KY.

[4] Aranda, J. and G. Venolia. The Secret Life of Bugs: Going Past the Errors and Omissions in Software Repositories. in *ICSE*. 2009. Vancouver, BC.

[5] Bass, M., J.D. Herbsleb, and C. Lescher. Collaboration in Global Software Projects at Siemens: An Experience Report. in *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*. 2007.

[6] Battin, R.D. and R. Crocker, Leveraging Resources in Global Software Development. *IEEE Software*, 2001. **18**(2): p. 8p.

[7] Bott, E., *Family and social network; roles, norms, and external relationships in ordinary urban families*. 2 ed. 1971, London,: Tavistock Publications. 252 p.

[8] Boyd, J.P., W.J. Fitzgerald, and R.J. Beck, Computing core/periphery structures and permutation tests for social relations data. *Social Networks*, 2006. **28**(2): p. 165-178.

[9] Bradner, E. and G. Mark. Why distance matters: effects on cooperation, persuasion and deception. in *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. 2002. New Orleans, Louisiana, USA: ACM.

[10] Carmel, E. and R. Agarwal, *Tactical Approaches for Alleviating Distance in Global Software Development*, in *IEEE Software*. 2001. p. 22-29.

[11] Carney, T.F., *Content analysis : a technique for systematic inference from communications*. 1972, Winnipeg: University of Manitoba Press. xx, 342.

[12] Cataldo, M., J.D. Herbsleb, and K.M. Carley. Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development. in *International Workshop on Socio-Techical Congruence, ICSE08*. 2008. Leipzig, Germany.

[13] Cataldo, M., et al. Identification of coordination requirements: implications for the Design of collaboration and awareness tools. in Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work. 2006 of Conference. Banff, Alberta, Canada: ACM.

[14] Cohn, B.S. and M. Marriott, Networks and centres of integration in Indian civilization. *Journal of Social Research*, 1958(1): p. 1-9.

[15] Conway, M.E., *How Do Committees Invent?*, in *Datamation*. 1968.

[16] Cubranic, D., *Hipikat*. 2004, University of British Columbia, IBM Ottawa Software Lab, and the National Research Council of Canada: British Columbia, Canada.

[17] Curtis, B., H. Krasner, and N. Iscoe, A field study of the software design process for large systems. *Commun. ACM*, 1988. **31**(11): p. 1268-1287.

[18] Damian, D., et al. Awareness in the Wild: Why Communication Breakdowns Occur. in *International Conference on Global Software Engineering*. 2007.

[19] Damian, D., I. Kwan, and S. Marczak, Requirements-Driven Collaboration: Leveraging the Invisible Relationships Between Requirements and People, in *CoSE*. 2009 To be appeared.

[20] Damian, D.E. and D. Zowghi, Requirements Engineering challenges in multi-site software development organizations. *Requirements Engineering Journal*, 2003. **8**: p. 149-160.

[21] Eclipse Foundation, *Eclipse*. 2009.

[22] Ehrlich, K., G. Valetto, and M. Helander. Seeing inside: Using social network analysis to understand patterns of collaboration and coordination in global software teams. in *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*. 2007.

[23] Everett, M.G. and S.P. Borgatti, Extending centrality, in *Models and Methods in Social Network Analysis*, P. Carrington, J. Scott, and S. Wasserman, Editors. 2005, Cambridge University Press: Cambridge.

[24] Freeman, L.C., Centrality in Social Networks Conceptual Clarification. *Social Networks*, 1978. **1**: p. 215-239.

[25] French, A. and P. Layzell. A Study of Communication and Cooperation in Distributed Software Project Teams. in Proceedings of the International Conference on Software Maintenance. 1998 of Conference: IEEE Computer Society.

[26] Frost, R., Jazz and the Eclipse Way of Collaboration. *Software, IEEE*, 2007. **24**(6): p. 114-117.

[27] Gamma, E. and J. Wiegand. The Eclipse Way: Processes that Adapt. in *Eclipse Con*. 2005.

[28] Gloor, P.A., et al. Visualization of Communication Patterns in Collaborative Innovation Networks - Analysis of Some W3C Working Groups. in Proceedings of the twelfth international conference on Information and knowledge management. 2003 of Conference. New Orleans, LA, USA: ACM.

[29] Griffin, A. and J.R. Hauser, Patterns of communication among marketing, engineering and manufacturing: a comparison between two new product teams. *Management science*, 1992. **38**(3): p. 360-373.

[30] Grinter, R.E., J.D. Herbsleb, and D.E. Perry. The geography of coordination: dealing with distance in R&D work. in Proceedings of the international ACM SIGGROUP conference on Supporting group work. 1999 of Conference. Phoenix, Arizona, United States: ACM.

[31] Gupta, A.K., Raj, S.P., Wilemon, D.L. , The R&D-marketing interface in high-technology firms. *Journal of Product Innovation Management*, 1985. **2**: p. 12-24.

[32] Herbsleb, J. and R. Grinter, *Architectures, Coordination, and Distance: Conway's Law and Beyond* in *IEEE Software*. 1999.

[33] Herbsleb, J.D. and R.E. Grinter. Splitting the organization and integrating the code: Conway's law revisited. in Proceedings of the 21st international conference on Software engineering. 1999 of Conference. Los Angeles, California, United States: IEEE Computer Society Press.

[34] Herbsleb, J.D. and A. Mockus, An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 2003. **29**(6): p. 14p.

[35] Herbsleb, J.D., et al. Distance, dependencies, and delay in a global collaboration. in Proceedings of the 2000 ACM conference on Computer supported cooperative work. 2000 of Conference. Philadelphia, Pennsylvania, United States: ACM.
[36] Herbsleb, J.D. and D. Moitra, *Guest Editors' Introduction: Global Software Development*, in *IEEE Software*. 2001. p. 16-20.
[37] Herbsleb, J.D., D.J. Paulish, and M. Bass. Global software development at siemens: experience from nine projects. in Proceedings of the 27th international conference on Software engineering. 2005 of Conference. St. Louis, MO, USA: ACM.
[38] Hinds, P. and C. McGrath. Structures that work: social structure, work structure and coordination ease in geographically distributed teams. in Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work. 2006 of Conference. Banff, Alberta, Canada: ACM.
[39] Hinds, P. and M. Mortensen, Understanding conflict in geographically distributed teams: An empirical investigation. *Organization Science*, 2005. **16**: p. 290-307.
[40] Hogenraad, R., D.P. McKenzie, and N. Peladeau, Force and Influence in Content Analysis: The Production of New Social Knowledge. *Quality and Quantity*, 2003. **37**: p. 221-238.
[41] IBM Inc. *Jazz Community Site*. 2006 [cited 2009 March 8]; Available from: http://jazz.net/.
[42] IBM Inc. *Jazz Team Wiki*. 2007 [cited 2007 May - December]; Available from: http://jazz.net.
[43] IBM Research, *Stellation*. 2003, IBM Research.
[44] Inkeles, A., *Soviet Reactions to the Voice of America*. 1952, Oxford University Press on behalf of the American Association for Public Opinion Research. p. 612-617.
[45] Intland Software, *codeBeamer*. 2009, Intland Software: Stuttgart, Germany.
[46] Kilduff, M. and W. Tsai, *Social networks and organizations*. 2003, London ; Thousand Oaks CA: SAGE Publications. 172.
[47] Kraut, R.E. and L.A. Streeter, Coordination in software development. *Commun. ACM*, 1995. **38**(3): p. 69-81.
[48] Li-Te, C., et al., *Building Collaboration into IDEs*. 2004, ACM. p. 40-50.
[49] Lings, B., et al., Ten Strategies for Successful Distributed Development, in *The Transfer and Diffusion of Information Technology for Organizational Resilience*. 2006. p. 119-137.
[50] Marczak, S., et al. Information Brokers in Requirement-Dependency Social Networks. in *International Requirements Engineering Conference*. 2008. Barcelona, Catalunya, Spance.
[51] Nguyen, T., A. Schroeter, and D. Damian. Mining Jazz: An Experience Report. in Infrastructure for Research in Collaborative Software Engineering. 2008 of Conference.
[52] Nohria, N. and R.G. Eccles, *Networks and organizations : structure, form, and action*. 1992, Boston, Mass.: Harvard Business School Press. xvi, 544.
[53] Olson, G.M. and J.S. Olson, Distance Matters. *Human-Computer Interaction*, 2000. **15**(2): p. 139 - 178.

[54] Perry, D.E., A.A. Porter, and L.G. Votta. Empirical studies of software engineering: a roadmap. in Proceedings of the Conference on The Future of Software Engineering. 2000 of Conference. Limerick, Ireland: ACM.

[55] Reuters (2009) QA Labs Earns a Ranking in the Top 10 Outsourced Software Testing & QA Vendors Report Two Years in a Row.

[56] Rulke, D.L. and J. Galaskiewicz, Distribution of Knowledge, Group Network Structure, and Group Performance. *Manage. Sci.*, 2000. **46**(5): p. 612-625.

[57] Sarker, S. and S. Sahay, *Implications of space and time for distributed work: an interpretive study of US-Norwegian systems development teams*. 2004, Macmillan Press Ltd. p. 3-20.

[58] Sheskin, D.J., *Handbook of parametric and nonparametric statistical procedures*. 2004, Boca Raton: Chapman & Hall.

[59] Shimbel, A., Structural parameters of communication networks. *Bulletin of Mathematical Bio-physics*, 1953(15): p. 501-507.

[60] Siegel, S., *Nonparametric statistics for the behavioral sciences*. McGraw-Hill series in psychology. 1956, New York,: McGraw-Hill. xvii, 312.

[61] The Mozilla Organization, *Bugzilla*. 2008.

[62] Treinen, J.J. and S.L. Miller-Frost, Following the sun: case studies in global software development. *IBM Syst. J.*, 2006. **45**(4): p. 773-783.

[63] Valetto, G., et al. Using Software Repositories to Investigate Socio-technical Congruence in Development Projects. in Proceedings of the Fourth International Workshop on Mining Software Repositories. 2007 of Conference: IEEE Computer Society.

[64] Wasserman, S. and K. Faust, *Social network analysis : methods and applications*. Structural analysis in the social sciences ; 8. 1994, Cambridge Cambridgeshire ; New York: Cambridge University Press. xxxi, 825.

[65] Wasserman, S. and K. Faust, *Social network analysis : methods and applications*. Structural analysis in the social sciences 8. 1994, Cambridge [Cambridgeshire] ; New York: Cambridge University Press. xxxi, 825 p.