

**Collaboration patterns of successful globally distributed
agile software teams: the role of core developers**

Sherlock Anthony Licorish

A thesis submitted to
Auckland University of Technology
in fulfilment of the requirements for the degree of
Doctor of Philosophy (PhD)

2013

School of Computing and Mathematical Sciences
Primary Supervisor: Professor Stephen MacDonell
Secondary Supervisor: Dr. Andy Connor

Table of Contents

List of Tables.....	iv
List of Figures	v
Attestation of Authorship.....	vii
Acknowledgements	ix
Abstract	xi
Chapter 1. Introduction and Background	1
1.1 Rationale for Research on Human Factors and Empirical SE.....	1
1.2 Main Motivations	2
1.3 Goals and Objectives.....	4
1.4 Scope, Assumptions and Boundaries	6
1.5 Research Design	8
1.6 Contributions and Published Work	12
1.7 Thesis Structure	15
1.8 Chapter Summary	16
Chapter 2. Literature Review	18
2.1 The Study of Human Factors.....	18
2.2 Team Composition, Psychology and SE Human Factors Frameworks.....	21
2.3 Globally Distributed Agile Software Development	24
2.4 The Study of Communication	26
2.5 Communication, Text, Language and Attitudes.....	29
2.6 Communication and SE Research	33
2.7 Wheel Structure Networks and Central Communicators	38
2.7.1 Attitudes and Team Roles	40
2.7.2 Changes in Attitudes and Knowledge Sharing.....	43
2.7.3 Attitudes, Knowledge Sharing and Task Performance	46
2.8 Chapter Summary	48
Chapter 3. Research Methodology and Design.....	50
3.1 Selecting a Research Method	51
3.2 Research Perspectives - Positivist versus Interpretivist	52
3.3 A Pragmatic Research Approach.....	54
3.4 Case Study Method and Study Design	56
3.4.1 Study Repository	61
3.4.2 Data Extraction and Pre-processing - Data Mining	65
3.4.2.1 Data Extraction and Pre-processing Procedures.....	68
3.4.3 Data Analysis (Technique 1) - Social Network Analysis	72
3.4.3.1 SNA and Other Quantitative Measures and Procedures	76
3.4.4 Data Analysis (Technique 2) - Linguistic Analysis	79
3.4.4.1 Behaviour and Attitude Analysis Measures	82

3.4.4.2	Linguistic Analysis Procedures	85
3.4.5	Data Analysis (Technique 3) - Content Analysis.....	86
3.4.5.1	Forms of Content Analysis.....	87
3.4.5.2	Reliability and Validity Issues in Content Analysis.....	89
3.4.5.3	Creating a Reliable and Valid Protocol.....	90
3.4.5.4	Selecting a Unit of Analysis.....	92
3.4.5.5	Content Analysis Tools	93
3.4.5.6	Ethical Requirements of Content Analysts	93
3.4.5.7	Content Analysis in SE and IS Research.....	94
3.4.5.8	Use of Content Analysis in this Study.....	94
3.5	Process of Theorizing.....	101
3.6	Chapter Summary and Methodological Framework	107
Chapter 4.	Results	110
4.1	Phase 1 – Social Network Analysis.....	110
4.1.1	Project Communication Patterns (RQ1)	110
4.1.2	Equity in Practitioners’ Communication (RQ2)	113
4.1.3	Importance, Task Performance and Formal Roles (RQ3, RQ4 and RQ5).....	119
4.2	Phase 2 – Linguistic Analysis and Directed CA (Static Analyses).....	125
4.2.1	Attitudes (RQ6).....	126
4.2.2	Enacted Roles (RQ7).....	130
4.3	Phase 3 – Linguistic Analysis and Directed CA (Longitudinal Analyses)	138
4.3.1	Attitudes (RQ8).....	141
4.3.2	Knowledge Sharing (RQ9) and Becoming Team Hubs (RQ10).....	144
4.3.3	Task Performance (RQ11)	149
4.3.4	Attitudes and Task Performance (RQ12).....	151
4.3.5	Knowledge Sharing and Task Performance (RQ13).....	153
4.4	Chapter Summary	154
Chapter 5.	Discussion	157
5.1	Collaboration patterns (Phase 1)	157
5.1.1	Communication patterns (RQ1).....	158
5.1.2	Equity in contribution (RQ2)	161
5.1.3	Active communicators importance (RQ3)	164
5.1.4	Active communicators task performance (RQ4).....	167
5.1.5	Active communicators formal roles (RQ5).....	169
5.1.6	Summary	172
5.2	The true role of core developers (Phase 2).....	175
5.2.1	Differences in attitudes (RQ6)	175
5.2.2	Enacted roles (RQ7).....	181
5.2.3	Summary	186
5.3	Changes in core developers’ attitudes, knowledge sharing and task performance (Phase 3).....	190

5.3.1	Changes in attitudes (RQ8)	190
5.3.2	Changes in knowledge sharing (RQ9)	192
5.3.3	Becoming team hubs (RQ10).....	195
5.3.4	Changes in task performance (RQ11)	196
5.3.5	Attitudes and task performance (RQ12)	197
5.3.6	Knowledge sharing and task performance (RQ13).....	198
5.3.7	Summary	199
5.4	Chapter Summary and Explanatory Model	201
Chapter 6. Conclusions		204
6.1	Retrospections	204
6.1.1	Collaboration patterns (Phase 1)	204
6.1.2	The true role of core developers (Phase 2).....	208
6.1.3	Changes in core developers' attitudes, knowledge sharing and task performance (Phase 3)	211
6.2	Research Contributions	214
6.2.1	Contributions to Theory	214
6.2.2	Contributions to SE Literature	217
6.2.3	Contribution to Pragmatic Research in SE.....	220
6.3	Research Evaluation, Limitations and Threats	222
6.4	Research Implications	228
6.4.1	Implications for SE Practice	228
6.4.1.1	Software Project Governance	228
6.4.1.2	Collaboration and Process Tools	232
6.4.2	Implications for SE Research (Future Work)	234
References		237
Appendices		270
Appendix I. Median message per WI communicated over project phases (P1- P10)		270
Appendix II. Sociograms for of all ten Jazz teams (P1 – P10)		271
Appendix III. Interaction behaviours (counts) for the UE, Code and PM project practitioners.....		272
Appendix IV. Percentages of interaction behaviours across the UE, Code and PM project areas		272
Appendix V. Combined percentages of overall project interaction behaviours for core developers		273
Appendix VI. Summary of project interaction for the core developers and others (for Code project area (P7))		273
Appendix VII. Summary of project interaction for the core developers and others (for PM project area (P8))		274
Appendix VIII. Descriptive statistics for core developers messages across the project phases		274
Appendix IX. Aggregated interactions for core developers.....		275
Appendix X. Confidentiality Agreement		276

List of Tables

Table 1. Constructivist versus Positivist research dichotomy.....	54
Table 2. Summary statistics for the selected Jazz teams.....	72
Table 3. Sample LIWC output variable information	81
Table 4. Sample MRC dictionary file	81
Table 5. LIWC linguistic measures.....	85
Table 6. Coding categories for measuring interaction	99
Table 7. Summary of research perspectives, questions and study techniques and measures.....	107
Table 8. Descriptive statistics for combined teams' (P1 – P10) communication	111
Table 9. Descriptive statistics for teams' in-degree measures (P1 – P10)	115
Table 10. Social network measures for core developers and their team scores (P1 – P10)	120
Table 11. Activities performed by core developers	124
Table 12. Descriptive statistics for linguistic scores for core developers and others ...	127
Table 13. Results for linguistics analysis	127
Table 14. Results comparing differences in selected language usage for core developers involved in multiple project areas	129
Table 15. Mean project area measures for messages, tasks, contributors and codes	132
Table 16. P1 (UE), P7 (Code) and P8 (PM) core developers' formal roles.....	136
Table 17. Numbers of messages communicated by core developers.....	140
Table 18. Descriptive statistics for core developers' linguistic measures across the project phases	143
Table 19. Counts of core developers' interactions (utterances).....	148
Table 20. Percentage of overall task (WI) changes made by core developers over the duration of their Project	150
Table 21. Summary of theoretical contributions.....	216
Table 22. Research evaluation taxonomy	222


List of Figures

Figure 1. Typical software risk items.....	21
Figure 2. Abstract representation of the wheel structure communication network	40
Figure 3. Case study model.....	57
Figure 4. Consolidated research questions.....	60
Figure 5. Components of the Jazz platform	62
Figure 6. Teams’ arrangement in Jazz	62
Figure 7. The data mining or knowledge mining process.....	66
Figure 8. Sample WI viewed via the RTC	68
Figure 9. Database model of the pre-processed and partially normalized Jazz data.....	70
Figure 10. Sociogram highlighting interaction patterns of team members.....	74
Figure 11. Section of Jazz communication network	75
Figure 12. Directed network graph for a sample Jazz team showing highly dense network segments for practitioners “12065” and “13664”	76
Figure 13. Extended database model after Linguistic processing.....	86
Figure 14. The Content Analysis protocol development process	91
Figure 15. Initial coding process.....	100
Figure 16. Simple visual basic interface that was created for entering codes into the Microsoft SQL database.....	100
Figure 17. Extended database model after Coding process	101
Figure 18. The methodological framework of this PhD	109
Figure 19. Mean messages per WI communicated over project phases (teams P1– P10)	113
Figure 20. In-degree measures for a sample Jazz team.....	114
Figure 21. Jazz sample team network (phases one (start) to four (end))	118
Figure 22. Summary of social network measures for the ten project areas (P1 – P10)	119
Figure 23. Sample network graphs (sociograms).....	123
Figure 24. Behaviour category (utterances) and number of occurrences for P1, P7 and P8	132
Figure 25. Percentages of overall team interaction (utterances) behaviours for the core developers	134
Figure 26. Summary of project interaction (utterances) for the core developers and others (for the UE team (P1)).....	135
Figure 27. Aggregated interactions (utterances) for core developers	146
Figure 28. Detailed interactions (utterances) of core developers over project phases..	147
Figure 29. Collaboration patterns of successful globally distributed agile software teams	174

Figure 30. Collaboration patterns of successful globally distributed agile software teams and the true role of core developers	189
Figure 31. Collaboration patterns of successful globally distributed agile software teams, the true role of core developers and changes in core developers' attitudes, knowledge sharing and task performance	203

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

A handwritten signature in black ink, reading "Sherlock Licorish". The signature is written in a cursive style with a long horizontal stroke at the end.

Sherlock Licorish

In loving memory of
my father, Victor Emanuel Saul.
Gone but not forgotten!

Acknowledgements

Multiple individuals have contributed directly and indirectly to this endeavour over the past three years. While some have provided direct enrichment throughout the course of this PhD, others have worked hard to remove obstacles, offered continuous encouragement and other forms of support. These contributions, in no small way, have led to the realisation of this thesis.

I am forever indebted to my primary supervisor, Professor Stephen MacDonell, who has done everything possible to ensure that this experience was inspiring and pleasant. Steve, your mentorship and guidance have provided the platform through which challenges are now frequently seen as opportunities, thank you. Thanks also to my secondary supervisor, Dr. Andy Connor, for guidance and advice.

I would like to express my gratitude to Dr. Jacqui Finlay, Frederik Schmidt and Ofosuhene Apenteng for their help and support at various stages of this journey.

Thanks to Associate Professor Tony Clear and Dr. Robert Wellington for their early critique and advice on the Confirmation of Candidature proposal. Others that have also instilled a desire for, and commitment to excellence, that have shaped my scholarly endeavours, and this PhD, particularly Jim Buchan, Anne Philpott, thank you.

Thanks to Professor Daniela Damian, Professor Didar Zowghi, Dr. James Skene and Dr. Jacqueline Whalley from whose suggestions and criticism multiple mental impressions have been formed over this journey.

I am very thankful to Dr. Kate Ehrlich of IBM Research for granting this work access to the IBM Rational Jazz repository. Thanks also to the many IBM Rational Jazz software practitioners for agreeing to release their development information from which the software engineering process may be properly investigated. I trust that the output of this research will inform your development practices.

I am grateful for the employment opportunities that were provided by SERL and SCMS, AUT, in the form of Lecturing, Project Supervision, Tutoring and Research Assistant roles. Thanks to Dr. Gisela Klette for the excellent BCIS R&D project coordination, and the students for their positive reviews and confidence – you’ve made it easy.

Thanks to my fellow colleagues at SERL, AUT, and particularly Amjed Tahir, Bilal Raza, Michael Bosu, Minjuan Tong and Waqar Hussain who have provided encouragement and support, and multiple forums through which I could share and receive knowledge along this journey.

It is with profound gratitude that I acknowledge the kind support of my family. I am forever grateful to my wife, Mondy, and son, IJ, for their patience, and for sacrificing their comforts in Guyana in support of my journey over the past three years. Thanks to my mom, Norma who has always stood behind me and knew I would succeed. Thanks to my sister, Deborah, who sowed the seed from which my competitive drive has grown.

I would like to express my sincerest appreciation to friends in Guyana, particularly Cleveland Sullivan, Lalchandra Rampersaud, Margaret Cummings, Mellisa Layne, Rawle Joseph and Shawn Croney, who have done an exceptional job removing various forms of obstacle, and providing assistance in order to limit the effects of possible interruptions on my study outcomes.

Finally, I am extremely thankful to the Vice Chancellor of AUT for his financial support through a Doctoral Scholarship award which has funded this PhD. Thanks also to the Minister of Public Service, Guyana, H.E. Dr. Jennifer Westford for her travel approval.

Abstract

Agile global software development (AGSD) has become increasingly prevalent in the last decade. AGSD occurs in environments that are inherently affected by elevated incidence of team-related challenges and issues, and so a growing body of research has centred on understanding teams that have overcome the challenges introduced by distance, and have succeeded in delivering high-quality products. Opportunities to study such teams are invaluable in terms of strengthening the AGSD knowledge base.

In particular, evidence captured in repositories and software logs, which are commonly used for communication in AGSD settings, has provided novel insights into team processes. One such observation is evidence of a centralised communication and task performance pattern for both open source and closed source AGSD teams. While this pattern has been noted, however, there has been little effort to understand *the reason behind* this phenomenon. Previous research has shown that central individuals are important to their teams' performance, as they coordinate information flow. Thus, understanding these members' roles in, and contributions to AGSD contexts would add significantly to the software engineering literature base considering the study of team dynamics from communication logs and repository data. The research reported in this thesis is a contribution to that literature base.

This research has used data mining techniques, social network analysis (SNA), psycholinguistic analysis and directed content analysis (CA) under a pragmatic case study design to study artefacts contributed by ten IBM Rational Jazz AGSD teams comprising 146 software practitioners, in order to explain the collaboration patterns of successful AGSD teams and how and why core developers contribute to AGSD team dynamics.

Drawing on role theories, behavioural and organisational psychology, social motivation, sociology and group interaction theories to delineate the novel findings from this case study, it is revealed that: communication patterns established early in a software project are maintained throughout; successful teams have highly connected communication networks; core communicators are also core developers; successful AGSD teams are social and task driven; formal role assignment does not pre-determine communication and coordination actions; and core developers operate across roles – both organisational and interpersonal. The results have shown that core developers' task performance

influences their need to communicate, and that core developers' performance is linked to the demands of their teams. Findings also established that core developers' language processes are related to their involvement in knowledge sharing and task performance.

These findings form the basis for initial conjectures of explanatory theories. This research also extends the software engineering literature and AGSD knowledge base, contributes insights for those intending to adopt a pragmatic approach in the study of software repository data, provides directions for further research, and outlines implications for software engineering practice.

In terms of software engineering practice, project managers are encouraged to plan for inevitable variations in communication volume, to consider task assignment as a mechanism to enhance knowledge redundancy and so reduce reliance on core developers, to exercise flexible project governance in order to facilitate self-organisation, to ensure a mix of social- and task-focused practitioners in AGSD teams, to encourage team-wide participation when core developers are most active to maximise team mentoring, to support core developers by surrounding them with other excellent communicators, and to facilitate frequent communication by top task performers. Emergent evidence in communication may also inform project diagnostics, and software tools with features that manage coordination requirements and prioritise team communication are likely to aid AGSD project governance and team performance.

Chapter 1. Introduction and Background

This chapter introduces the work that has been conducted for this PhD and is reported in this thesis. In the first section (Section 1.1), the rationale for this study is provided, including the justification for conducting empirical software engineering (SE) research, and particularly, for studying human factors in SE - the focus of this work. The specific motivations underpinning this research are provided in Section 1.2, including the author's personal motivation which was driven by anecdotal evidence and experiences gained while working in multiple software practitioner roles, and gaps identified in the research literature. Details of the study's goals and the more finite objectives are provided in Section 1.3, and the study scope, assumptions and boundaries are outlined in Section 1.4. The research design is described in Section 1.5, where the research epistemology, methods and techniques are briefly introduced. A summary of the overall contributions of the work is provided in Section 1.6, including individual aspects that have resulted in specific published works. An outline of the thesis structure is provided in Section 1.7, and finally, Section 1.8 provides a summary of the details provided in this chapter.

1.1 Rationale for Research on Human Factors and Empirical SE

Contention over software systems' adequacy, project success rates and the adoption of appropriate software process models have been ubiquitous and longstanding (Augustine, Payne, Sencindiver, & Woodcock, 2005; Boehm, 2006; Siddiqui & Hussain, 2006). Despite many recommendations in relation to specific software methodologies and tools (Chin, 2004; Licorish, Philpott, & MacDonell, 2009b; Nerur, Mahapatra, & Mangalaraj, 2005), there remain questions over the outcomes of software development projects (Charette, 2005; Standish Group, 2009). Previous evidence suggests that people factors, such as communication issues and behavioural conflicts, underscore the causes of collaborators' (dis)satisfaction and inadequately performing software teams (Abrahamsson et al., 2006; Acuna, Gomez, & Juristo, 2009; Rajendran, 2005; Verner & Evancho, 2005; Walle & Hannay, 2009). Thus, it is a widely held view that studying these topics would provide a suitable avenue through which researchers could understand the software process and could offer (more) valuable recommendations for process improvements.

In fact, beyond specific recommendations for more people-centric consideration, over the last decade empirical research and evidence based software engineering (EBSE) has become fundamental to the understanding of software development practices and technologies (Benestad & Hannay, 2012; Heijstek, Kuhne, & Chaudron, 2011; Jaanu, Paasivaara, & Lassenius, 2012; Siniaalto & Abrahamsson, 2007; Wohlin, 2002). The view of proponents in the empirical software engineering space is that software development practices and technologies are context-specific (Bird, Murphy, Nagappan, & Zimmermann, 2011), and only through empirical avenues can these contextual understandings emerge so that particular practices and technologies are leveraged appropriately to provide value to the software engineering community. Outcomes and understandings from empirical investigations often lead to teams' empowerment and provide insights and awareness for software development stakeholders (Bird et al., 2011; Franca & Da Silva, 2009).

For instance, Zimmermann et al. (2010) investigated the features that made bug reports most helpful to developers and found that steps to reproduce bugs, stack traces and test cases are most helpful for locating defects during bug fixing; this information, however, was rarely provided by testers. In terms of the potential utility of this finding for software project governance, this observation may encourage testers to provide the regularly omitted information in their reports which would likely reduce the time developers spend attempting to replicate bugs, and may also inform additional requirements for bug reporting tools. Failure prediction studies have also allowed software teams to predict risky software features and prioritise testing, and have aided the creation of plans for future maintenance activities (Bhat & Nagappan, 2006; Bird et al., 2011; Nagappan & Ball, 2007). Studies have also considered under which circumstances developers should work together given their need to coordinate and communicate around specific software features (Bird, Nagappan, Gall, Murphy, & Devanbu, 2009; Pinzger, Nagappan, & Murphy, 2008), providing team composition recommendations. This latter theme is particularly relevant to this work, and in part, motivates this study agenda, as considered in the next section (Section 1.2).

1.2 Main Motivations

Group dynamics and group maturity processes generally, and more specifically as they apply to software development, provide the stimulation and curiosity that drive this research project. Having had the exposure and privilege to work as a Programmer for

over four years and as an Information Systems Manager for three years, this researcher has gained first-hand insights into how individuals and their roles change over time when they are organised into or form groups. During this process it was noted that deliverables from groups are affected by individuals' (changing) behaviours over the software project's lifecycle. Additionally, it was observed that individuals' interactions and involvement in groups were linked to both their personal motivation and public persona. Other personal group experiences, derived from involvement in the corporate world, academia and social networks, have also informed this researcher's interest in and convictions about the complex and subtle contributions that individuals and group dynamics bring to software development.

Such has been the keen interest in and attention to the details of groups that this researcher is almost certain that there exist patterns in group dynamics that are necessary for, and that can be intelligently harnessed towards, ensuring that software projects succeed. If found, such patterns could be distilled and synthesised into models to support the management of team behaviours and group dynamics based on rigorous empirical research evidence. Such models would be especially useful for globally distributed agile software developments given that such teams are often challenged with reduced levels of awareness, group identification and shared understandings, due to team members' separation (Chang & Ehrlich, 2007; Jaanu et al., 2012).

Given these impediments, issues related to behavioural imbalance are likely to have a negative impact on global teams' performance, and particularly given the way negative behavioural incidents are shown to affect both trust and team spirit (Feldt, Angelis, Torkar, & Samuelsson, 2010). In fact, because of the growing attention given to global and distributed software development, with many major market players such as Microsoft, IBM and Oracle using this approach during the delivery of noteworthy software releases (Yu, Ramaswamy, Mishra, & Mishra, 2011), it is imperative to examine successful globally distributed software teams to understand the behavioural configurations under which these teams perform best. Additionally, such teams are said to adopt agile software development practices (Coram & Bohner, 2005), which are inherently driven by people processes.

Recent research studies have placed significant emphasis on studying the communication and coordination practices of software engineering teams, as noted in the following examples: (Cataldo, Wagstrom, Herbsleb, & Carley, 2006; Damian,

Marczak, & Kwan, 2007; Ehrlich & Chang, 2006). Behavioural issues have also attracted noteworthy attention in the literature due to the way they are posited to impact individuals' ability to engage during team work and their willingness and capacity to communicate and form/sustain relationships in teams (Acuna & Juristo, 2004; Feldt et al., 2010; Karn & Cowling, 2006; Wynekoop & Walz, 2000). However, little research effort has been dedicated to studying the intricacies of globally distributed agile software team dynamics, and particularly how core developers in these settings influence their team's performance. Therein lies an avenue for the work that is performed here, the goals and objectives of which are presented next in Section 1.3.

1.3 Goals and Objectives

The primary goal of this work is to understand and explain the collaboration patterns of successful globally distributed agile software teams, and how and why core developers contribute to globally distributed agile team dynamics. These understandings should extend previous theories and provide team composition, task assignment and management recommendations for software project leaders. Since human actions are performed within the confines of social systems, and such social systems are rooted in specific controlled norms (Giddens, 1979), the more finite goals (the research questions that address these goals are formally represented in Section 1.5) of this study are to unearth how teams' communication patterns change during a successful globally distributed agile software project, to understand how core software developers operate within structural teams, and how these teams operate as agents within organisational structures. This latter goal involves explaining the true role of core developers and how their attitudes, behaviours and task performance impact their project's health as their project progresses.

Such understandings and recommendations are likely to be most constructive if they are garnered from highly effective and successful globally distributed agile software teams. Of course, a more complete understanding would be provided by also considering failed projects; however, data from such projects are not often available. While particular mechanisms to solicit insights and experiences, such as interviews and questionnaires (Damian, Izquierdo, Singer, & Kwan, 2007), may contribute towards the achievement of this research goal, the outputs of this work would be most meaningful if they are derived from the examination of teams' artefacts used/produced during the execution of an actual successful globally distributed agile software development project. This latter

requirement is particularly necessary in this study context because software practitioners are known to provide socially desirable responses when completing questionnaires and answering interview questions (Holden & Passey, 2010). Additionally, these individuals are likely to operate within normative organisational boundaries when outsiders are present (Goguen, 1993). Furthermore, is it also generally infeasible to gain access to globally distributed agile software practitioners during software developments due to the geographical spread of such teams. Thus, this work inspects software artefacts using a multi-phase mixed method approach, to examine globally distributed agile software teams' dynamics during the delivery of successful software outcomes.

To address the goal outlined above, the specific objectives of this study are as follows:

1. To examine the way communication patterns of successful globally distributed agile software teams change over the course of their project and to explore the collaboration patterns of such teams, including the way team members' interactions and task performance are distributed.
2. To establish how the most influential globally distributed agile software practitioners (core developers) contribute to team dynamics; this includes how these individuals contribute to actual software development activities and problem solving while occupying their given roles.
3. To establish the true role of core developers, including how these members contribute their social and intellectual capital, whether their attitudes differ from those of other practitioners, and the actual roles they enact during a globally distributed agile software development project.
4. To examine how the core developers contribute their social and intellectual capital as their project advances; this includes examining whether core developers' attitudes change as their project progresses, how core developers share knowledge over the course of their project, the initial team arrangements that lead to core developers becoming hubs in their teams, the way core developers contribute to task performance over the course of their project, the way core developers' contributions to task performance is linked to their attitudes and how core developers' contributions to task performance is linked to their contribution of knowledge.

5. To advance theory for understanding human processes during software development and to inform studies mining software repositories; to provide models and methodological advice and encourage future work examining human processes through the use of software artefacts (refer to the following section (Section 1.4) for further details).

1.4 Scope, Assumptions and Boundaries

As explained above, this work aims to contribute to research and practice in globally distributed agile software development. Given that globally distributed software teams operate in dispersed contexts, these individuals often use tools to support their communication and coordination activities (Yu et al., 2011). In fact, some globally distributed software development teams have shown a preference for communicating with text-based tools (e.g., email, blogs and instant messaging) as against through video, audio and face-to-face mechanisms (Jaanu et al., 2012; Yu et al., 2011). Such tools often provide persistent data storage, and so, are valuable sources of interaction evidence from which software development human processes could be studied (Abreu & Premraj, 2009; Bachmann & Bernstein, 2009). The utility provided by these repositories is in fact noteworthy as research evidence has shown that a substantial amount of developer time is spent on communication (Perry, Staudenmayer, & Votta, 1994). Additionally, the unobtrusive nature of gaining access to repository artefacts also provides researchers with added novelty. Answers related to why specific steps are taken during the software development process, and how decisions are implemented, are often evident in such artefacts, and particularly if the communication and collaboration tools used during the software development process are the primary mechanism for project coordination and control (Nguyen, Wolf, & Damian, 2008). As indicated in Section 1.3, this research uses repository data to study globally distributed software teams' human processes. Accordingly, this work is conducted in conformance with the following assumptions and boundaries:

1. This work assumes that the artefacts studied from the selected repository are central to the coordination and management of the SDLC (Nguyen, Wolf, et al., 2008), and communication evidence in these software artefacts reveals how, what and why software practitioners communicate (Damian, Izquierdo, et al., 2007; Singer, 1998) – from both behavioural and knowledge perspectives.

2. Although concepts from the hermeneutic theory of Ricoeur (1981) could be used to artificially construct the physical environment from the virtual environment (see further discussions on the application of Ricoeur (1981)'s work in Klein & Myers (1999)), there is no such attempt in this project, nor is there any physical involvement between the author of this work and the software practitioners being studied.
3. This work assumes that theories and frameworks in management and role theories (Belbin, 2002; Benne & Sheats, 1948), behavioural and organisational psychology (Colomo-Palacios, Cabezas-Isla, Garcia-Crespo, & Soto-Acosta, 2010; Downey, 2009), social motivation (Geen, 1991; Inkpen & Tsang, 2005; Levin & Cross, 2004), sociology (Hackman, 1986) and group interaction and psycholinguistics (Pennebaker, Chung, Ireland, Gonzales, & Booth, 2007; Pennebaker & King, 1999), provide adequate grounding for determining attitudes, behaviours and knowledge sharing from artefacts.
4. This study examines artefacts that are contributed by many of the typical software practitioners (programmers, team leaders, project managers, administrators, and those occupying multiple roles) that are commonly involved during software development. However, many of the findings and recommendations are aimed at those responsible for software project governance (software project managers and team leaders). The teams studied in this work are globally distributed and are adopting specific agile practices through which multiple software outputs have been successfully deployed (Frost, 2007). These software systems are commercially accessible and were verified by clients as adequately usable. Details around the clients' feedback are available at <http://www.jazz.net>.
5. Given the incremental development stance inherent in agile methodologies (Coram & Bohner, 2005), the individual project phases, although easily identifiable in the repository, may not be consistent from team to team. However, the research strives to assemble and consider all of the artefacts in the repository that belong to individual teams, and the project is partitioned in a realistic way that closely reflects the way a software project occurs over time.
6. Cultural differences and distance (geographical and temporal) may directly affect software development teams' performance (Espinosa, DeLone, & Lee, 2006), and these variables may also have an impact on team members' behaviours – which in turn may lead to performance issues (Jaanu et al., 2012). However, research

examining the effects of cultural differences in global software teams has found few cultural gaps and behavioural differences among software practitioners from, and operating in, Western cultures, with the largest negative effects observed between Asian and Western cultures (Espinosa et al., 2006). Given that the teams studied in this work all operated in Western cultures, this work does not consider the culture and distance dimensions of globally distributed agile software development, and the way these might affect individual behaviours. Rather, this work focuses on the social attitudes and knowledge contribution of software practitioners and how these variables interplay during team work – by considering practitioners, their roles and responsibilities, their teams’ structure and size, their teams’ task portfolio, and the organization.

7. While the previous experiences of the author (discussed in Section 1.2) may potentially bias the analyses and interpretations provided throughout this work, the author’s past involvement with the software development process, and software artefacts in general, are also likely to strengthen these analyses and interpretations. Additionally, these experiences are likely to reinforce the reliability and validity of this work. Further, in accordance with the recommendation of previous theories, where necessary, the author’s views and assessments are validated with established theories in the relevant field (Neuendorf, 2002; Weber, 1990).

1.5 Research Design

In order to address the research objectives and provide the contributions outlined above, this work utilises a pragmatic approach (Newman & Benz, 1998), employing a mixed method case study design (refer to Chapter 4 for further details). Some aspects of the work outlined above are confirmatory and quantitative in nature, and these are best studied under the guise of a positivist framework (Onwuegbuzie & Leech, 2005), whereas other aspects of the work are exploratory and are driven in a bottom-up fashion by a more interpretivist or constructivist approach (Klein & Myers, 1999). Given that this work uses repository data, quantitative approaches are employed for data reduction, data cleaning and analysis of the quantitative data in the early part of the work (Onwuegbuzie & Leech, 2005). The more qualitative aspects of the work are guided by thematic analysis techniques, using a bottom-up approach, towards the provision of initial theories (Onwuegbuzie, 2003). Quantitative measures are then used to identify

meta-themes and relationships among themes discovered through quantitative and qualitative observations (see the work of Barcellini, Detienne, Burkhardt, & Sack (2008) for example). These approaches are utilised to provide multiple insights and strengths to the work under consideration here (Leech & Onwuegbuzie, 2009; Schultz & Hatch, 1996). By using quantitative techniques to analyse themes revealed from qualitative data analysis this study provides deeper levels of interpretation from this aspect of the work. Additionally, qualitative aspects of the work help to explain (Easterbrook, Singer, Storey, & Damian, 2008; Leech & Onwuegbuzie, 2009) statistically significant findings revealed during the quantitative elements of the work, as a means of providing triangulation for the methods and techniques selected (refer to Chapter 4 for further details).

Given the case study design and the research objectives, this study utilises multiple units of analysis, at the organisation, team, and individual levels (Runeson & Host, 2009). In the first research phase, data mining principles (Han & Kamber, 2006; Tan, Steinbach, & Kumar, 2006) are applied to the data collection and preliminary data exploration activities. During this phase quantitative data analysis of the repository is conducted to select appropriate cases and to uncover preliminary insights around how globally distributed agile software teams' communication changes over the course of the software project. SNA techniques (De Laat, Lally, Lipponen, & Simons, 2007; Willging, 2005) are also used to enable confirmatory analyses to be conducted during this phase. This form of analysis is used to compare quantitative findings in this work with those reported in previous studies (Mockus, Fielding, & Herbsleb, 2002; Nguyen, Wolf, et al., 2008). Data analyses conducted through the use of SNA also inform the case selection process (Crowston, Wei, Li, & Howison, 2006) and later explorations of core developers' behaviours and attitudes, and their knowledge sharing and task performance processes. The research questions that are outlined to address these objectives are (refer to Chapter 2 for further details):

RQ1. Do communication patterns change as the software project progresses?

RQ2. Is there equity in practitioners' contributions to their project?

RQ3. Are active communicators more important to their teams' collaboration?

RQ4. How are active communicators involved in task performance?

RQ5. Are practitioners' formal role assignments related to their involvement in project interactions and task performance?

Previous research has established that a few individuals in each team generally dominate project communication and source code changes during software development (Bird, Gourley, Devanbu, Gertz, & Swaminathan, 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009). Additionally, previous work has shown that core developers occupy the centre of their teams' information sharing actions, and they are (therefore) critical to team performance (Bavelas, 1950). These members have also been shown to influence their wider teams' willingness to adapt to change and maintain performance (Ruhnow, 2007). However, questions related to *the reasons for* these members' extraordinary presence, and understanding the actual roles (both formal and informal) that core developers occupy in their teams, have not been answered. As noted in Section 1.6, such answers could be invaluable for understanding the nature and peculiarities of globally distributed agile software teams' dynamics and informing the process of assembling high-performing and cohesive teams.

In the second project phase of this research, extracted communication data is analysed using linguistic analysis and directed content analysis techniques (Henri & Kaye, 1992; Pennebaker et al., 2007; Pennebaker & King, 1999; Zhu, 1996). These approaches are used in a more exploratory manner to study core developers' behaviours and attitudes and their enacted roles (Belbin, 2002). The following research questions are designed to address these issues (refer to Chapter 2 for further details):

RQ6. Do core developers' behaviours and attitudes differ from those of other software practitioners?

RQ7. What are the core developers' enacted roles in their teams, and how are these roles occupied?

Similar to the analysis threads in the second phase just mentioned, linguistic analysis (Pennebaker et al., 2007; Pennebaker & King, 1999) and directed content analysis (Babbie, 2004) are conducted in a third research phase to study changes in core practitioners' attitudes and knowledge processes (Henri & Kaye, 1992; Zhu, 1996) and the way these relate to these members' actual contribution to software development activities. Core developers maintain exceptional performance in both team communications and task changes (Cataldo & Herbsleb, 2008; Shihab et al., 2010). However, it is not clear how these individuals contribute to their teams' process over the

course of their project, and how their organizational, interpersonal, intrapersonal and management competencies sustain their project's health. Additionally, there is uncertainty around what team conditions, and over which project phase(s), core developers are most important to their teams. Previous work has shown that practitioners' interaction patterns change over their project (Cataldo & Ehrlich, 2012; Cataldo & Herbsleb, 2008; Cummings & Cross, 2003), and so longitudinal studies should uncover details that could lead to explanations for software team dynamics more fully. Evidence of how practitioners interact over the course of their project will inform targeted team strategies and phase-specific interventions. In fact, previous calls for such investigations of team dynamics have been made (Hinds & McGrath, 2006), as the static view does not reveal fully what actually happens over the duration of a software development project. Accordingly, the following questions are designed to study changes in core developers' behaviours and attitudes and knowledge sharing:

RQ8. Do core developers' attitudes change as their project progresses?

RQ9. How do core developers share knowledge over the course of their project?

RQ10. What initial team arrangements lead to developers becoming hubs in their teams?

Knowledge sharing studies have shown that individuals' and teams' willingness to actively participate in knowledge sharing and team performance is linked to multiple factors (Hinds & Pfeffer, 2003), and particularly social motivation (Inkpen & Tsang, 2005; Levin & Cross, 2004). Accordingly, core developers may be most happy to perform during periods of positive and social behavioural climate (De Vries, Van den Hooff, & De Ridder, 2006; Zakaria, Amelinckx, & Wilemon, 2004). Similarly, core developers may also exhibit eagerness to perform during highly evaluative periods. Studying these members' attitudes and knowledge sharing behaviours and the way these are linked to their involvement in their teams' task performance would provide further insights into globally distributed agile software teams' dynamics. Thus, the following questions are also answered in the third phase of this research (refer to Chapter 2 for further details):

RQ11. How do core developers contribute to task performance over their project?

RQ12. Are core developers' contributions to task performance linked to their attitudes?

RQ13. Are core developers' contributions to task performance linked to their contribution of knowledge?

As noted above, each research phase builds on those that have preceded it, but it is also common for patterns observed in a later phase to inform reviews and enhancements of activities conducted in an earlier phase (e.g., analyses outputs of the SNA inform a review of the cases that are selected during the preliminary data mining explorations). In summary, during this multi-phase analysis, the research questions (RQ1 – RQ13) outlined above are answered under the guidelines of the case study method and general principles of pragmatism. The contributions that are derived from these explorations are presented in the next section (Section 1.6).

1.6 Contributions and Published Work

In achieving the goals and objectives outlined in Section 1.3 above, this work provides multiple contributions to both software engineering theory and practice. From a theoretical perspective, this work extends research focusing on understanding human processes during software development, and particularly, the body of literature that has revealed evidence of how teams work through the use of software artefacts and repository data (Ehrlich, Helander, Valetto, Davies, & Williams, 2008; Ehrlich, Lin, & Griffiths-Fisher, 2007). This study also provides methodological advice and recommendations for those extracting and examining software repository data (Nguyen, Wolf, et al., 2008; Shihab, Bettenburg, Adams, & Hassan, 2010; Shihab, Zhen Ming, & Hassan, 2009), and provides an extended taxonomy which offers an avenue through which others may understand globally distributed agile software team dynamics and core software practitioners' attitudes and knowledge sharing behaviours (Licorish & MacDonell, 2013a, 2013c). From a research design perspective, through this work researchers are likely to gain grounded insights into pragmatism (Newman & Benz, 1998) and the implementation of multi-method research (Licorish & MacDonell, 2013a). Through theoretical lenses, secondary contributions of this work include pointers and actual exemplars of the implementation of tested approaches from the organisational psychology, management and role theories and social psychology disciplines (Licorish & MacDonell, 2013a). Further, this work also provides specific recommendations for future research, with a view to advancing software engineering theory and providing additional understandings and advice for software development practice (Licorish & MacDonell, 2013b, 2013c).

From a practice-based perspective, this work provides the following concrete understandings and recommendations aimed at improving globally distributed agile software team composition and task assignment processes:

1. Grounded evidence of, and explanations for, how globally distributed agile software teams' communications change over the course of their project and details around the collaboration patterns of such teams, including the way team members' interactions and software changes are distributed (Licorish & MacDonell, 2013c, 2013d).
2. Grounded evidence of, and understandings for, how the most active globally distributed agile software practitioners (core developers) contribute to team dynamics, particularly including explanations for how these individuals contribute to actual software development activities and problem solving while occupying given roles (Licorish & MacDonell, 2013b, 2013c).
3. Explanations of how core developers contribute their social and intellectual capital, how their attitudes differ from those of other practitioners, and the actual roles they enact during a globally distributed agile software project (Licorish & MacDonell, 2013c) – these understandings are aimed at establishing why a centralised pattern is seen for software teams' communication and explaining the true role of core developers.
4. Understandings of the way core developers' attitudes change as their project progresses, how core developers share knowledge over the course of their project, the initial team arrangements that lead to core developers becoming hubs in their teams, the way core developers contribute to task performance over their project, the way core developers' contributions to task performance are linked to their attitudes, and how core developers' contributions to task performance are linked to their contribution of knowledge – these understandings are aimed at informing project staffing and specific team configurations in support of the most active software practitioners during the software development process.
5. Recommendations for extending collaboration and process support tools (Licorish & MacDonell, 2013c, 2013d).

Through these understandings and recommendations software project leaders will be informed about how to plan for the staffing of globally distributed agile software teams. Project leaders and software engineering stakeholders will also understand the particular

characteristics of distributed agile group dynamics, how to assemble global teams with appropriate behavioural configurations, and how to identify ‘software gems’ – exceptional practitioners both in terms of task and team performance. Knowledge of the means by which the most active practitioners become the centre of their project, and the way attitudes and knowledge sharing behaviours are linked to these software developers’ task performance, would help project leaders to identify software standouts and aid with assembling high-performing and cohesive teams. Suggestions for new tool features are also aimed at improving the software development process.

While this thesis presents a consolidation of the aforementioned contributions, more granular outputs are contributed in the following published works:

1. Licorish, S.A., & MacDonell, S.G. Understanding the attitudes, knowledge sharing behaviors and task performance of core Jazz developers: A longitudinal study, Under second review with Information and Software Technology.
2. Licorish, S. A. and MacDonell, S. G. (2013) Self-organising roles in agile globally distributed teams, in Proceedings of the 24th Australasian Conference on Information Systems (ACIS 2013). Melbourne, Australia, ACIS, pp.TBC.
3. Licorish, S.A., & MacDonell, S.G. (2013) Adopting softer approaches in the study of repository data: a comparative analysis, in Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE2013). Porto de Galinhas, Brazil, ACM Press, pp.240-245. doi:10.1145/2460999.2461035.
4. Licorish, S.A., & MacDonell, S.G. (2013) Differences in Jazz project leaders’ competencies and behaviors: a preliminary empirical investigation, in Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). San Francisco CA, USA, IEEE Computer Society Press, pp.1-8. doi: 10.1109/CHASE.2013.6614725.
5. Licorish, S.A., & MacDonell, S.G. (2013) How do globally distributed agile teams self-organise? Initial insights from a case study, in Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE2013). Angers, France, SCITEPRESS, pp.227-234. doi: 10.5220/0004437001570164.

6. Licorish, S.A., & MacDonell, S.G. (2013) The true role of active communicators: an empirical study of Jazz core developers, in Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE2013). Porto de Galinhas, Brazil, ACM Press, pp.228-239. doi:10.1145/2460999.2461034.
7. Licorish, S.A., & MacDonell, S.G. (2013) What can developers' messages tell us? A psycholinguistic analysis of Jazz teams' competencies and behavior patterns, in Proceedings of the 22nd Australasian Software Engineering Conference (ASWEC2013). Melbourne, Australia, IEEE Computer Society Press, pp.107-116. doi:10.1109/ASWEC.2013.22.
8. Licorish, S.A., & MacDonell, S.G. (2012) What affects team behavior? Preliminary linguistic analysis of communications in the Jazz repository, in Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). Zurich, Switzerland, IEEE Computer Society Press, pp.83-89. doi:10.1109/CHASE.2012.6223029.

1.7 Thesis Structure

This thesis comprises six main chapters. The current chapter (Chapter 1) introduces the thesis and provides background information about the work that is conducted during this research project. Chapter 2 provides a survey of literature, including the rationale for studying human factors, an introduction to theories around team composition, psychology and software engineering human factors frameworks, a review of globally distributed agile software developments and the study of communication, an outline of the way attitudes are revealed from textual communication and a comprehensive review of studies dedicated to the subject of communication in the software engineering domain. This latter review has led to the identification of several research gaps (also briefly introduced in Section 1.5 above), and accordingly, an outline of 13 research questions (refer to Section 2.6 and Section 2.7).

Chapter 3 presents the research methodology and design, including a review of the principles for selecting a specific research method or conforming to a particular research epistemology (positivist and interpretivist), an introduction to the pragmatic approach employed in this work, an outline of the case study method and the techniques that are utilised during this work and a review of the theorising process. The process of

theorising particularly emphasises the way in which explanation theories are provided through empirical investigations, the focus of this work (refer to Section 3.5).

Chapter 4 presents the results of this work. These results are set out in three main sections (Section 4.1, Section 4.2 and Section 4.3) in alignment with the incremental research design introduced in Section 1.5 above. A similarly incremental approach is used in the research discussion which is presented in Chapter 5. This thesis then concludes with Chapter 6, which provides retrospections of the study results and discussions, an outline of the study's contributions, an evaluation of this research project's limitations and threats, and a summary of the work's implications. Aspects of the work not captured in the six main chapters (Chapter 1 – Chapter 6), particularly additional illustrations and statistics, are included in the Appendices.

1.8 Chapter Summary

This chapter presented the introduction and background to this thesis. In Section 1.1 it was noted that previous evidence suggests that investigations aimed at understanding the human processes involved during software development are likely to provide value for the software development community, perhaps much more so than those studies aimed at providing tool support and software methodologies. Additionally, it was shown that empirical software engineering studies would provide important contextual explanations for software process improvements. This work is shown to conform to this framework, and also employs an empirical approach to study globally distributed agile software teams' human processes. Section 1.2 noted that the drive to study human issues is particularly fitting given the way such concerns are exacerbated in distributed software development settings, due to reduced opportunities for team engagement.

Section 1.3 introduced the research goals of this work - to understand and explain the collaboration patterns of successful globally distributed agile software teams, and how and why core developers contribute to globally distributed agile software team dynamics. This goal was decomposed into multiple granular sub-goals, and further, into five study objectives in Section 1.3. These objectives are addressed through the study of software artefacts as contained in a software repository. Studying this medium is especially appropriate for this work given that globally distributed software teams use this means for project communication, and such sources have been held to capture the details of the software development process once it is used as (one of) the primary means of communication.

In this regard, seven assumptions and boundaries were formulated in Section 1.4, and the contributions of this work are assessed in this regard. The research design of this work was introduced in Section 1.5, which showed that this thesis embraced both positivism and interpretivism under the general principles of a pragmatic case study design. Section 1.5 also briefly introduced 13 research questions; these are confirmatory and exploratory, and underpin the pragmatic approach that is adopted in this study. The work's contributions were outlined in Section 1.6, and comprise those for software engineering theory (i.e., for theoretical advancement, from a design perspective and identification of avenues for future research), and software development practice (recommendations for project governance and extending collaboration and process support tools). Finally, an outline of the thesis structure is presented in the preceding section (Section 1.7). This section indicates that this thesis comprises six main chapters, which are also supported by a range of Appendices.

The second of these chapters (Chapter 2) is presented next, providing an advancement of this research agenda. Chapter 2 considers the study's literature review, and provides an in-depth examination of the theories in support of this work.

Chapter 2. Literature Review

This chapter critically reviews prior research and considers various theories that are relevant to the research themes addressed in this work. First, since this study considers teams' and individuals' involvement during the software development process, Section 2.1 sets out the motivation for studying this issue. Section 2.2 then surveys research on team composition, considering this subject from the perspective of sociology and behavioural psychology. This section (Section 2.2) also considers the study of teams in software engineering, and introduces the most relevant software engineering human factors frameworks. The next section (Section 2.3) introduces the specific context of globally distributed agile software development, providing explanations of how agile practices are implemented by globally distributed software teams. This section (Section 2.3) also considers the particular relevance of communication in a globally distributed agile software development setting. The following section (Section 2.4) outlines the benefits that are gained in studying communication, introduces different communication methods, and provides an introduction to the literature around software engineering teams' communication. Section 2.5 then reports theories regarding how attitudes are revealed in communication, with particular emphasis on the way attitudes are demonstrated in textual exchanges – the environment that is examined in this study. Works dedicated to the study of software practitioners' communication artefacts such as electronic messages, change request histories, bug logs and blogs are then surveyed in Section 2.6, and preliminary research questions are outlined at this stage. This survey (in Section 2.6) reveals that the wheel structure communication pattern is evident for most software teams, and so this structure and its implications are considered in detail in Section 2.7. Through this detailed assessment, further research gaps are identified and appropriate research questions are outlined throughout Section 2.7. This chapter then comes to a close in Section 2.8 with a brief summary of the theories presented throughout the preceding sections.

2.1 The Study of Human Factors

Software development involves interdependent individuals working together to achieve favourable outcomes both in terms of team productivity and product quality. This establishes the long-held nature of development as a team effort (Walz, Elam, & Curtis, 1993). However, since the emergence of software development as a discipline it has been plagued with contradictions over the adoption of specific procedures and tools, as

well as inconsistencies in terms of projects' success rates (Abrahamsson et al., 2006; Boehm, 2006; Licorish, Philpott, & MacDonell, 2009a; Siddiqui & Hussain, 2006; Standish Group, 2009). These outcomes result in extensive speculation over which approaches and tools are more suitable, when, and for whom, and how they should be used to provide maximum value for the software development community (Boehm, 2006; Boehm & Turner, 2003a, 2003b; Chin, 2004; Licorish et al., 2009b).

Nonetheless, despite on-going efforts to improve software development practices through such initiatives, uncertainties over project success rates remain (El Emam & Koru, 2008). This has led to a growing belief that software development performance would improve more substantively if the human processes employed during this activity were better understood and supported (Abrahamsson et al., 2006; Acuna et al., 2009; Beranek, Zuser, & Grechenig, 2005; Capiluppi, Fernandez-Ramil, Higman, Sharp, & Smith, 2007; Capretz & Ahmed, 2010; Cunha, Canen, & Capretz, 2009; Rajendran, 2005).

In fact, almost irrespective of the reports of software project failures (Standish Group, 1995, 2001), studying human-related issues would seem to be necessary given the emphasis placed on individuals and interactions (Coram & Bohner, 2005) and collaboration and coordination (Chang & Ehrlich, 2007) by the increasingly prevalent software development approaches in use in both agile and global software development settings (Abrahamsson et al., 2006; Acuna et al., 2009; Cataldo et al., 2006; Damian, Izquierdo, et al., 2007; Damian, Kwan, & Marczak, 2010; Ocker & Fjermestad, 2008; Rajendran, 2005).

In light of the above, there is a growing body of research studies dedicated to human interaction, communication and coordination themes (Abreu & Premraj, 2009; Al-Rawas & Easterbrook, 1996; Cataldo & Ehrlich, 2012; Damian, Izquierdo, et al., 2007; De Vries et al., 2006; Ehrlich & Cataldo, 2012; Hayes Huffman, 2003; Herbsleb, Mockus, Finholt, & Grinter, 2001; Howison, Inoue, & Crowston, 2006; Licorish & MacDonell, 2012). This phenomenon is also aligned with the consensus of recent evidence, which continues to indicate that a variety of human and social factors are among the strongest determinants of software development project performance (e.g., refer to Abrahamsson et al.(2006) for details).

Outcomes of research on software risk management (Paul, 2008; Ropponen & Lyytinen, 2000; Schmidt, Lyytinen, Keil, & Cule, 2001; Zwikael & Ahn, 2010) converge with those in mainstream software engineering regarding the relevance of human factors in software project performance (Abrahamsson et al., 2006; Cataldo & Ehrlich, 2012; Denning, 2012; Sach, Sharp, & Petre, 2011; Sharma & Kaulgud, 2011; Zhou & Mockus, 2011). This convergence is highlighted in Figure 1, which presents the main problem areas (risks factors) for software development (Paul, 2008; Ropponen & Lyytinen, 2000; Schmidt et al., 2001; Zwikael & Ahn, 2010). As noted in Figure 1 (adapted from <http://www.myglobalit.com/>), while some risks relate directly to human issues (e.g., “coordination issues” and “team conflict”), others may be perceived to be less so (e.g., “inadequate testing” and “changing requirements”). However, on close examination of the model in Figure 1, it may be deduced that regardless of the technology or tool support, people factors stand at the centre of this activity (a concept illustrated in Figure 1). In fact, even “technology issues”, or the oversight of these, relate fundamentally to people and their conduct, whether through inadequate reporting or the neglect of suitable contingencies.

Accordingly, studying the people processes involved during software development should provide fruitful avenues for researchers to better understand software development practices, and to offer recommendations for software project governance and overall process improvements. The benefits of such research are evident in the body of work dedicated to the study of software practitioners’ communication and coordination practices, which has provided noteworthy understandings and recommendations for software project control (Cataldo et al., 2006; Damian, Izquierdo, et al., 2007; Damian et al., 2010; Ocker & Fjermestad, 2008). Software practitioners’ behavioural issues (e.g., personality and trust) have also attracted notable attention in the literature. These works have offered suggestions for dealing with the impact of individuals’ behaviour traits during team work (Feldt et al., 2010; Gallivan, 2001; Karn & Cowling, 2006; Wynekoop & Walz, 2000; Zheng, Veinott, Bos, Olson, & Olson, 2002).

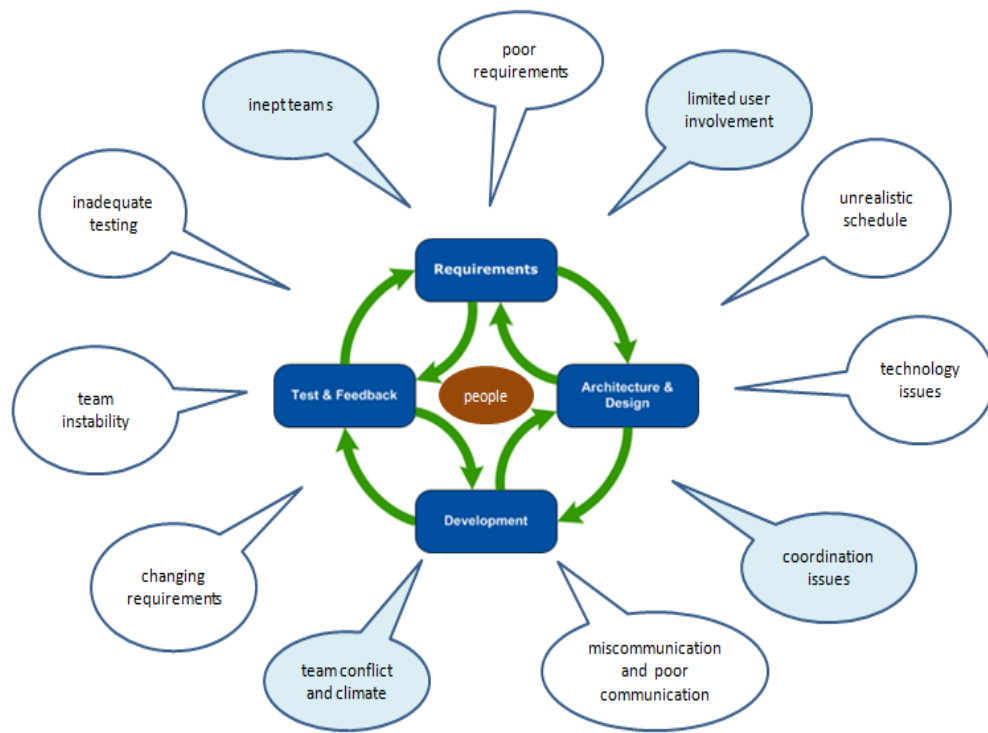


Figure 1. Typical software risk items

This study follows this line of research, and draws on social network analysis psycholinguistics and directed content analysis to study multiple aspects of globally distributed agile teams' behavioural dynamics. Given that this work uses principles from psychology to study this subject, psychology and the most relevant software engineering human factors frameworks are reviewed in the following section (Section 2.2) to provide contextual understanding for how these disciplines contribute to software human resource management. This material is reviewed prior to introducing the extant literature in the globally distributed agile software development space (provided in Section 2.3).

2.2 Team Composition, Psychology and SE Human Factors Frameworks

Team composition and individuals' social and behavioural influences during group-work are said to impact the outcomes of group tasks (Hackman, Morris, & Leonard, 1975; Zalesny, 1990). These issues have been considered from many perspectives, including sociology and behavioural psychology relating to self-identity (Thomas & Hynes, 2007), social identity (Blaskovich, 2008; Brown, 2000), social capital (Oh, Labianca, & Chung, 2006), group emotion and group mood (Gummer, 2001; Smith, Seger, & Mackie, 2007), emotional intelligence (Druskat & Wolff, 2001), and team building (Katzenbach & Smith, 2001). According to the resultant theories, each

individual brings a unique set of knowledge and skills to their collaboration during group work, which is influenced by participants' social and behavioural qualities (Watson & Michaelson, 1988). Such qualities are said to determine how team members interact and the likelihood of teams being cohesive and productive (Adams & Anantatmula, 2010).

As a consequence, human resource management has leveraged psychology and management theories in supporting the task of selecting individuals with appropriate skill sets for positions (Beranek et al., 2005; Capretz & Ahmed, 2010; Pollock, 2009; Stevens & Henry, 1997). In particular, although job advertisements in the software development industry generally emphasise technologies (Colomo-Palacios et al., 2010; Litecky, Arnett, & Prabhakar, 2004), most software development related positions demand multiple capabilities, including intrapersonal, organisational, interpersonal and management skills (e.g., see monster.com) (Acuna, Juristo, & Moreno, 2006; Colomo-Palacios et al., 2010; Downey, 2009). Intrapersonal skills include judgement, innovation and creativity and tenacity, while being self-organising and having knowledge of specific environments (e.g., programming competences in Java or Microsoft technologies) is characterised as organisational (Colomo-Palacios et al., 2010; Downey, 2009). Interpersonal skills comprise team work and cooperation and negotiating skills, and management skills are related to planning, organisation and leadership (Acuna et al., 2006). In relation to software groups or departments, roles may also relate to the specific software process or methodology being utilised by teams (Downey, 2009). For instance, a software department adopting Extreme Programming may define roles such as programmer, tester, coach and so on (Highsmith, 2004). Additionally, sometimes software roles may be performed arbitrarily by project members in which case these environments require that team members possess general competency in many roles (Capretz & Ahmed, 2010; Gorla & Lam, 2004). Thus, role arrangement and competency requirements for individual software-related roles are somewhat subject to specific organisational requirements and contexts (Acuna et al., 2006; Trigo et al., 2010).

Apart from the consideration of capabilities in the human resource management area, and in particular, the specific application of such capabilities to software positions, software engineering as a discipline has also considered human involvement in software development activities. For instance, the Software Engineering Institute (SEI) provides the People-CMM to support the human dimensions of software development (Curtis,

Hefley, & Miller, 2001), the People Software Process (PSP) focuses on software participants' performance (Humphrey, 1997), the Soft System Methodology (SSM) considers software organisations' social systems (Checkland, 2000) and the Team Software Process (TSP) (Humphrey, 1998) provides improvement guidance for software teams. Overall, these models are all aimed at informing software practitioners' development, and augmenting the process of skills and capabilities management and software role assignment.

As noted above, agile methodologies also emphasise the people element in software development (Beck, 2000; Chin, 2004; Cockburn & Highsmith, 2001; Pressman, 2009). However, it has been argued that some of the benefits that derive from their use are eroded when agile is implemented in a global software development context (the environment studied in this work), due to communication barriers, lack of group identification, and trust and culture issues that are introduced by team members' separation (Chang & Ehrlich, 2007; Kamaruddin, Arshad, & Mohamed, 2012; Serce et al., 2009). Given these impediments, consequences related to teams' collaboration dynamics, attitudes and knowledge behaviour imbalances are likely to have a negative impact on global teams' performance. This is particularly relevant for global software developments given that unbalanced team configurations are said to affect overall team performance and team spirit (Andre, Baldoquin, & Acuna, 2011; Feldt et al., 2010).

Given the growing attention given to global and distributed software developments, with many market leaders such as Microsoft, IBM, Lucent and Oracle using this approach during the delivery of major software releases (Herbsleb et al., 2001; Yu et al., 2011), it is imperative to examine global software development teams to understand the dynamics of successful teams, and the behavioural configurations under which these teams perform best (Chang & Ehrlich, 2007; Serce et al., 2009). Understandings from these conditions and outcomes would provide pointers for project governance that others could seek to replicate.

Accordingly, this work systematically examines multiple issues under the team dynamics umbrella. The first necessity in this regard is an understanding of the way agile software development is performed in globally distributed settings. To this end, literature that addresses globally distributed agile developments is reviewed in the next section (Section 2.3).

2.3 Globally Distributed Agile Software Development

Geographically distributed work is becoming ubiquitous due to globalisation (Cataldo & Herbsleb, 2008); and this trend has found favour in some software development organisations (Bird, Nagappan, Devanbu, Gall, & Murphy, 2009; Herbsleb & Moitra, 2001; Nguyen, Wolf, et al., 2008). For instance, India's software industry grew between 30% and 40% annually for the ten year period ending in 2004 due to their involvement in global software ventures (Arora & Gambardella, 2005). Driven by the availability of cheaper hardware, affordable software development talent pools, increased access to communication infrastructure and technologies and the need to reduce the time-to-market, many software companies have expanded and are growing their operations to reach global markets (Karolak, 1999; McDonough, Kahn, & Barczak, 2001; Yu et al., 2011). In keeping with this expansion, these companies are employing global software development (GSD) approaches, where software teams operate across distances, including over continents and national boundaries (Layman, Williams, Damian, & Bures, 2006; Sahay, 2003). In some software development environments, GSD teams work in a distributed manner within relatively close proximity; however, as with distributed developments across national boundaries, they are unable to communicate face-to-face on a regular basis (Layman et al., 2006).

The drive to deliver software releases in rapid succession in order to reduce the time-to-market and gain greater market share demands that companies employing GSD use iterative software development methodologies. Thus, agile methodologies are often adopted by GSD organisations (Danait, 2005; Layman et al., 2006; Young & Terashima, 2008), in an approach also referred to as Agile Global Software Development (AGSD).

Compared to the relatively recent emergence of GSD, the concept of agility in software development has been in existence for three decades (Koch, 2005). This software development approach emphasises four values (as captured in the Agile manifesto): individuals and interaction over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan (Coram & Bohner, 2005). Additionally, agile proponents operate under 12 guiding principles (Koch, 2005). These values and principles are said to deliver value to software organisations and customers, while also leading to improved software quality and reduced risk.

In agile software development methodologies such as Extreme Programming (XP), Scrum, the Crystal Families of Methodologies, Feature-Driven Development, Adaptive Software Development, and Agile Modelling, there is a gradual surfacing of the software design and requirements, which promotes a human-centric environment, having persons interacting in a common space, employing a ‘speculate-collaborate-learn’ approach (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003).

Although many success stories have been reported regarding the implementation of agile methodologies in global software development contexts (or AGSD) (Danait, 2005; Layman et al., 2006; Young & Terashima, 2008), this phenomenon has also been reported to be quite challenging (Kamaruddin et al., 2012). In particular, team member dispersion (including customers and clients (Earl, 1996)) in AGSD has been shown to reduce the opportunities for informal (and face-to-face) communication (Carmel & Agarwal, 2001; Cataldo, Bass, Herbsleb, & Bass, 2007; Herbsleb & Mockus, 2003a; Jalali & Wohlin, 2010). This dispersion has also been shown to affect project oversight and monitoring (Damian & Zowghi, 2003; Prikladnicki, Nicolas Audy, & Evaristo, 2003; Rudzki, Hammouda, Mikkola, Mustonen, & Systä, 2010), and temporal distance has been reported to have a negative impact on team culture and trust (Dullemond, Gamenen, & Solingen, 2009; Lee & Yong, 2010). Changing requirements, and the consequent need for team (re)orientation to maintain the shared understanding that characterises agile software development, has also been deemed counterproductive for AGSD (Prikladnicki et al., 2003; Tiwana, 2004). On the other hand, the delivery of frequent software releases that is encouraged in co-located agile settings (Jones, 1996; Larman & Basili, 2003) is also recommended in ASGD environments (Paasivaara & Lassenius, 2003). Thus, AGSD project stakeholders are often required to be extremely vigilant and skilled to maintain team performance (Al-Ani, Horspool, & Bligh, 2011; Young & Terashima, 2008).

Due to the way AGSD teams operate in a distributed manner, individual team members often rely heavily on communication technologies to support their team processes (Abreu & Premraj, 2009; Bachmann & Bernstein, 2009; Yu et al., 2011). Of the many risk items introduced above, this issue (project communication) in particular, is often critical to AGSD teams’ performance (Carmel & Agarwal, 2001; French & Layzell, 1998; Herbsleb & Mockus, 2003a; Herbsleb et al., 2001; Kamaruddin et al., 2012; Layman et al., 2006). Given that team communication is often recorded for persistence in AGSD settings, such communications form a source that could provide novel details

into the software development process (Abreu & Premraj, 2009; Bachmann & Bernstein, 2009). The rationale for project decisions, pointers for how AGSD teams work, insights into the way such teams collaborate, and general details on AGSD team dynamics are stored in distributed software teams' communication logs. Thus, these logs provide invaluable knowledge-bases for AGSD (Bachmann & Bernstein, 2009). Such knowledge is particularly valuable when communication artefacts of successful teams are examined, and these teams use such artefacts as one of the primary source of team communication.

This work uses a sample of such artefacts to study globally distributed agile software team dynamics and, in particular, the attitudes, roles and knowledge sharing behaviours of core developers. The next section (Section 2.4) considers theories associated with the study of communication (and particularly communications that are text-based) in order to highlight the relevance of studying this subject. More generally, Section 2.4 provides understandings of the different forms of communication media and the variables that are used to measure their effectiveness.

2.4 The Study of Communication

Effective communication is critical to the outcomes of interactions, and good communication is recognised as essential for building interpersonal relations; these assertions are known to hold across numerous contexts. This intricate social phenomenon is defined as the ability to exchange information and express and comprehend thoughts, feelings and attitudes derived from those exchanges (Ivancevich & Matteson, 2001). The various methods for communicating enhance the complexities of this activity. Communication may occur in face-to-face settings, via addressed and unaddressed written forms or over electronic communication media such as telephone, video or computer-mediated channels.

Social presence and media richness are the variables commonly used when assessing the effectiveness of different communication channels. According to Short, Williams, & Christie (1976), social presence is used to categorise a communication medium based on the level of physical presence that is conveyed during participants' communication. In the measurement of physical presence, Rice (1992) and Short et al. (1976) highlight that timing, pauses, inflections, and non-verbal cues are significant attributes for effective communication. Non-verbal cues are facial expressions, gazes, posture and physical distance. Johansen (1977) and Reid (1977) also explain that social presence

may be measured by how communal (personal or warm) individuals feel while communicating.

On the other hand, media richness refers to the capability of those communicating to understand information conveyed via one or more channels in a specified timeframe (Daft & Lengel, 1986). Rich communication transactions are able to overcome different frames of reference and allow for clarification of issues in order to enable timely change to communicating participants' understanding. Therefore, communications that impede or delay communicators' understanding possess a lower degree of richness. Media richness may be measured by a medium's capability for instant feedback, the number, nature and diversity of cues available to those communicating, the language options available and the degree to which attention is focused on the communicators (Daft & Lengel, 1986).

In view of this, when assessing the theories regarding presence introduced by Johansen (1977) and Rice (1992), it may be inferred that non-verbal cues are significant indicators of social presence. Thus, face-to-face communication would be higher in presence when compared to video, telephone and computer-mediated communication (CMC) such as text chat and email. In this regard, the opportunity for greater self-awareness, reduced inhibition, and rapid responsiveness should be facilitated in face-to-face settings, followed by video, telephone, text chat and email (in that order). Hence, it would be rational to deduce (in principle) that interpersonal interaction would be most effective in a setting of high social presence and media richness.

While face-to-face communication indeed provides an optimal environment for rich communication and high levels of social presence during individual exchanges, text and telephone communications also provide individuals with effective ways to communicate and coordinate during teamwork. More than that, such forms of communication may also afford individuals an opportunity to express themselves (Dabbish, Kraut, Fussell, & Kiesler, 2005; Kennedy, McComb, & Vozdolska, 2011) without the need or desire to self-regulate and temper their views and opinions. Early studies considering small groups, which are common in agile software development, have provided general support for this position. Prashant (1997) posited that individuals' general willingness to conform diminishes in computer chat and email settings due to reduced normative pressure. The findings of Smilowitz, Compton, & Flint (1988) and Walther & Burgoon

(1992) also revealed less individual conformity and a greater willingness to self-disclose in computer-mediated settings.

While the preceding discussion considers communication in a general way, the effects of social presence (or lack thereof) and media richness on communication and team outcomes have been studied specifically in the software engineering and information system domains. Cummings (2004), Herbsleb (1999) and Herbsleb, Mockus, Finholt, & Grinter (2000) have all shown that lack of social presence and media richness affect team members' ability to communicate effectively, and group performance overall. In contrast, more recent work reported by Nguyen, Wolf, et al. (2008) found that these issues provided no barrier to the team members they studied when conducting group tasks. Bird et al. (2009) also found negligible differences in the incidence of software failures between distributed and collocated software development sites. In their study of an agile global software development team Layman et al. (2006) found that textual communication was considered to provide adequate means for project stakeholders to maintain knowledge sharing and information transfer over the duration of the project, and this method effectively substituted face-to-face communication. These latter results could be interpreted to suggest that more recent communication technologies are providing more effective support for software teams' communication processes (Calefato, Gendarmi, & Lanubile, 2009; Cheng et al., 2003; Frost, 2007), or that over time software teams have developed and exhibited higher levels of creativity and communication skills (Hall, Wilson, Rainer, & Jagielska, 2007; Lee, Trauth, & Farwell, 1995). That said, although there are mild disagreements concerning the modes and methods of communication, and how these affect communication efficacy, the overall benefits of communication cannot be overstated (Chang & Ehrlich, 2007).

Communication, regardless whether computer-mediated or face-to-face, signals team members' interaction and information exchange, and exploring these activities provides an avenue to understand the intricate nature of *actual* group work (Damian, Izquierdo, et al., 2007; Singer, 1998). For instance, it was discovered in previous research that developers' interactions during software projects are not a replication of those stated in the development plan (Cataldo et al., 2006; Damian, Izquierdo, et al., 2007).

Additionally, empirical evidence has shown that software artefacts and software history data are useful sources of interaction evidence (Aranda & Venolia, 2009). More specifically, it has been posited that communication artefacts such as electronic messages, change requests histories and blogs would provide a unique perspective on

activities occurring during the software development process (Cataldo et al., 2006; Singer, 1998). As noted in Section 2.3, such a position is especially valid if these artefacts are used as the *only* means of interaction during software development. Such deductions provide general support for the work undertaken and reported in this study. This work examines software teams' collaboration and behavioural issues as are evident through artefacts. Accordingly, the next step is to survey the application of appropriate theories to guide this activity, and to inform the selection of an appropriate approach for use in this work. This activity is considered in the following section (Section 2.5).

2.5 Communication, Text, Language and Attitudes

According to established linguistic theories it is possible to discern attitudes within individuals' communications (Mairesse, Walker, Mehl, & Moore, 2007; Pennebaker & King, 1999; Pennebaker & Lay, 2002; Pennebaker, Mehl, & Niederhoffer, 2003). Linguistic studies have shown that a person's language use is stable over periods of time, and that the way individuals communicate is influenced by their context and local settings (Mehl & Pennebaker, 2003). Language use has also been studied as a function of age (Pennebaker & Stone, 2003), gender (Mulac, Bradac, & Gibbons, 2001) and emotional upheavals (Stone & Pennebaker, 2002). Earlier works examining language use have also supported the viewpoint that there are unique variations in individuals linguistic styles from situation to situation and linguistic analysis of textual communication can reveal much about those communicating (Giles & Wiemann, 1993; Hart, 1984; Oxman, Rosenberg, Schnurr, & Tucker, 1988; Pennebaker & King, 1999; Pennebaker, Mayne, & Francis, 1997; Schnurr, Rosenberg, & Oxman, 1992; Spence, Scarborough, & Hoff Ginsberg, 1978; Taylor, Reed, & Berenbaum, 1994). These studies provide further compelling evidence that language use is contextual. So, for instance, an individual is likely to express happiness and satisfaction in their communication if they are fulfilled, while the opposite may be observed if they are dissatisfied (Stone & Pennebaker, 2002).

These attitudes may be easily detected during face-to-face interactions (Funder & Colvin, 1988; Gosling, Rentfrow, & Swann, 2003) where a high level of social presence exists. However, in circumstances where there are lower levels of social presence (e.g., in textual communication environments), the reliability of such assessments may be challenged (Blackman, 2002; Gill, Oberlander, & Austin, 2006). Particularly, in textual settings, researchers engaging in behavioural assessment using linguistic cues have

reported mixed results (Gill & Oberlander, 2002; Hancock & Dunham, 2001). Gill & Oberlander (2002) found behaviour patterns in text syntax, and these authors were also able to link individuals' personality traits to linguistic patterns when examining email communication. In contrast, and so less compelling, Hancock & Dunham (2001) reported a lesser degree of confidence in judging individual attitudes when studying linguistic features in internet chat, due to the reduced amounts of impressions available for analysis in text.

Support for the Gill & Oberlander (2002) findings was also provided in Gill et al. (2006)'s later study. Other works considering individuals' attitudes from a personality perspective have provided support for linking behaviours to linguistic patterns. For instance, Gill et al. (2006) conducted an experiment in which judges, after being briefly introduced to personality theories, rated subjects' personality following a short examination of their written text. Strong and significant correlation was found between judges' prediction and subjects' actual personality preferences. In particular, social attitudes were very pronounced during these linguistic observations. This same study found very little correlation between judges' perceptions of, and participants' actual attitudes for individuals exhibiting Neuroticism (displaying negative traits), however. Gill et al. (2006)'s findings for Neuroticism are also supported by the earlier results (Gill & Oberlander, 2002) where Neuroticism was not as clearly detected in textual communication when compared to attitudes that denoted Conscientiousness and Agreeableness.

This low agreement for some attitudes is perhaps understandable given that the human judges in these studies considered multiple language categories in each study subject's writing to derive a measure of personality (Gill et al., 2006; Hancock & Dunham, 2001). In contrast, works examining individual linguistic dimensions to assess precise attitudes have reported consistency between specific language use and individual traits (Giles & Wiemann, 1993; Mehl & Pennebaker, 2003; Pennebaker et al., 2003; Taylor et al., 1994). For instance, previous research has found elevated use of first person plural pronouns (we) during shared situations and among individuals that share close relationships, whereas relatively high use of self-references (e.g., I) has been linked to individualistic attitudes (Pennebaker et al., 2003; Stone & Pennebaker, 2002). Thus, individuals' conversations reflect their internal feelings (Denning & Dunham, 2010). Moreover, these traits are revealed in communication, regardless of the settings.

Although the evidence just provided does not mean that studying individuals' attitudes based on their textual communications is not without its challenges, and particularly when the goal is to capture the meaning, motive and context around words usage (Krauss & Fussell, 1996; Zeldow & McAdams, 1993), it is reasonable to deduce that text environments may provide communication avenues for those less likely to participate in 'real' (face-to-face) settings. These individuals are likely to subconsciously express their true feelings, and textual settings are likely to result in individuals' reduced desire to self-regulate and temper their views and opinions (Dabbish et al., 2005; Kennedy et al., 2011). In fact, because social individuals have a strong desire to be sociable and are assertive and dominant (Trapnell & Wiggins, 1990), their traits will likely result in their willingness to be expressive, humorous, persuasive, and verbose, regardless of the environment. Conversely, the more cognitive and conscientious individuals are said to be purposeful, achievement-oriented and organised (McCrae & Costa, 1987), and are generally anticipated to communicate less, and their patterns of communication are expected to be planned and schedule-oriented, task-focused and less verbose. As such, even in textual and low intensity communication environments, these (differences in) attitudes may be easily detected. Thus, studying textual communications should provide a rich means for observing individuals' attitudes.

As textual communication is a primary conduit for software teams' knowledge-sharing, particularly in agile globally distributed software development contexts (Jaanu et al., 2012; Yu et al., 2011), and given that developers' communication have been found to be correlated with software development activities (Shihab et al., 2010), studying the details of software practitioners' textual communications during the performance of various software development tasks should provide understandings of software team dynamics. In particular, evidence for the way various types of team traits are promoted as necessary for effective team collaboration could be validated through these means (Chang, Yen, Chiang, & Parolia, 2013; Denning, 2012).

To this end, data repositories and archives recording software developers' textual communication activities have been used extensively to study software practitioners' social behaviours (Ducheneaut, 2005; Mockus et al., 2002). While text analysis tools have been used previously to understand and predict various aspects of software development (Junior, Mendonca, Farias, & Henrique, 2010; Mockus & Votta, 2000; Shihab et al., 2009), only a few studies in this domain have considered examining

attitudes from developers' textual communication. At the time of this review (which covered searches in the ACM Digital Library, IEEE Xplore, EI Compendex, Inspec, ScienceDirect and Google Scholar) studies were discovered examining language use in relation to group member dominance (Zhou, Burgoon, Zhang, & Nunamaker, 2004), automatic personality recognition from speech (Polzehl, Moller, & Metze, 2010), behavioural perception in human agents (Prabhala & Gallimore, 2005), and the linkage of personality traits to posture and gesture (Ball & Breese, 2000). However, very few studies were found to formally apply approaches from the behavioural sciences and psycholinguistic space (Pennebaker et al., 2003; Stone & Pennebaker, 2002) to examine the details in software developers' communication logs (Rigby & Hassan, 2007).

Some researchers have studied the details of developers' textual communication, using a range of informal and mathematically-based approaches. For instance, Shihab et al. (2009) used tag clouds to represent what GTK+ and Evolution open source software (OSS) developers communicated about during their projects and noted that conversations around bugs and patches dominated the discussions of both groups, and specific development topics emerged for each team given the particular project focus during the different project phases. Shihab et al. (2009) also found that the top developers (those that communicated the most and made the most commits) referred to others by their actual names (directly addressing contributors), an observation that was linked to their status in the team. Shihab et al. (2010) found higher levels of use of certain terms to be associated with specific development activities (e.g., the terms "new feature" and "feature request" were found to be correlated with code addition, and words including "patch" and "testing" correlated with code modifications). Mockus & Votta (2000) used text analysis to classify changes as corrective or perfective in CVS logs. Li et al. (2006) combined manual classification, text analysis and other machine learning techniques to study bug descriptions.

Apart from the outputs produced in the course of this work (Licorish & MacDonell, 2012, 2013d), Rigby & Hassan (2007) are the only other researchers to have studied the details of software attitudes from developers' textual communications using formal approaches from the psycholinguistic space. Through the use of psycholinguistic analysis Rigby & Hassan (2007) revealed that once the top two developers signalled their intentions to leave the Apache project their mailing list communications became more negative and instructive, and they spoke mostly in the future tense and communicated with less positive emotions, when compared to their earlier

communications. Their study also found variations in communication behaviour after releases. In studying two releases Rigby & Hassan (2007) found that developers' communication was optimistic after the first release, whereas the opposite was evident after the second release. Such findings are insightful but are 'one-off' and point to the need for further exploratory research.

While questions in relation to reliability and validity may be posed for studies examining open source mailing lists, due to the way participants' communications are managed in this environment (i.e., anyone is able to post messages and report bugs to such mailing lists) (Bachmann & Bernstein, 2009), studies such as that of Rigby & Hassan (2007) have provided sufficiently useful discoveries to encourage linguistic analysis of developers' communication in other controlled environments.

Findings from such observations would provide useful insights about the software development process. These issues are considered further in the next two main sections (Section 2.6 and Section 2.7). In the first instance, Section 2.6 considers a larger survey of the literature around software teams' communication with a view to identifying research gaps and some preliminary research questions for exploring this phenomenon during a successful AGSD project. Section 2.7 then extends the literature assessment in terms of teams' centralised communication patterns in Section 2.6, and identifies further research gaps to precisely outline the remaining research questions of this work.

2.6 Communication and SE Research

Previous research has established that the intricacies of team dynamics can be revealed by studying members' communication¹ (Aranda & Venolia, 2009; Cataldo et al., 2006; Rigby & Hassan, 2007; Singer, 1998). Research has also exposed linkages between informal hierarchical communication structures and team performance for geographically distributed teams (Hinds & McGrath, 2006). Furthermore, team communication has been linked to coordination efficiency (de Souza & Redmiles, 2009) and the quality of software output (Herbsleb & Roberts, 2006). Thus, as noted in Section 2.4 and Section 2.5, studying the details in team communication can provide valuable insights into the human processes involved during software development,

¹ The terms "communication" and 'interaction' are used interchangeably throughout this thesis to mean the exchange of information.

including the reasons for, and consequences of, communication and coordination actions.

Given this, software repositories and software history data have emerged as valuable sources of interaction and communication evidence (Aranda & Venolia, 2009; Bird, Gourley, Devanbu, Gertz, & Swaminathan, 2006b; Cataldo & Ehrlich, 2012; Crowston, Wei, Howison, & Wiggins, 2008; Datta, Kaulgud, Sharma, & Kumar, 2010; Datta, Sindhgatta, & Sengupta, 2011; Ehrlich & Cataldo, 2012; Nguyen, Wolf, et al., 2008; Shihab et al., 2009). Research findings from works examining such sources can be considered particularly valid if the data that is examined represent the primary means of interaction and team processes during software development – other validity-related factors notwithstanding (Licorish & MacDonell, 2013c). Accordingly, previous researchers have exploited process artefacts such as electronic messages, change request histories, bug logs and blogs to provide unique perspectives on the activities occurring during the software development process (Cataldo et al., 2006; Singer, 1998). Particularly, OSS repositories and archives recording software developers' textual communication activities provided researchers with early opportunities to study practitioners' behaviours (Ducheneaut, 2005; Mockus et al., 2002), with recent work tending also to examine commercial repositories (Cataldo & Herbsleb, 2008). A selection of these studies is now considered.

Bird, Gourley, Devanbu, Gertz, & Swaminathan (2006a) employed clustering algorithms to study CVS records and mailing lists and concluded that the more software development an individual does the more coordination and controlling activities they must undertake. The Debian mailing list was used by Sowe, Stamelos, & Angelis (2008) to observe knowledge sharing among developers. These authors found that no specific individual dominated knowledge sharing activities in the Debian project. Abreu and Premraj (2009) observed the Eclipse mailing list and found that increases in communication intensity coincided with higher numbers of bug-introducing changes, and that developers communicated most frequently at release and integration time.

Crowston, Annabi, Howison, & Masango (2004) provided recommendations for using team effectiveness and coordination theories to study artefacts from Sourceforge. Crowston, Wei, Li, & Howison (2006) examined core developers of five small OSS projects using multiple explanatory approaches, including Bradford's law, and found that the core group of developers comprised only a small number of the total

contributors. Crowston & Howison (2006) used SNA to verify the social structure of 122 OSS projects in the Sourceforge repository, 22 projects from the GNU Savannah system and 56 projects from the Apache Software Foundation Bugzilla bug tracking system. These authors found some OSS projects to be highly centralised, and this pattern was especially pronounced for smaller projects. Additionally, it was revealed that most of the OSS projects in these three repositories had a hierarchical social structure, although there was more communication modularity in larger projects (Crowston & Howison, 2006).

Pohl & Diehl (2008) used SNA to study artefacts produced by four developers of the TOMCAT project and noted that developers' engagements overlapped for development and documentation activities. Using the GTK+ and Evolution OSS projects Shihab et al. (2009) also established that only a small number of developers participated in internet relay chat (IRC) meetings. Similarly, Shihab et al. (2010) found communication activity to be correlated with software development activity when studying the GNOME project, where what was communicated was reflected in source code changes. Shihab et al. (2010) observed that the most productive developers contributed 60% of the project's communication, and their interaction levels remained stable over the project duration when compared to lesser contributing participants. Yu et al. (2011) also found similar patterns of communication when studying artefacts' from the GNOME GTK+ project. However, these authors caution about the limitation of the frequency-based analysis approaches that they employed and recommended the use of techniques for studying variations in developers' actual language processes. Goguen (1993) had also previously argued for the use of sociolinguistics to study artefacts in order to understand human-related issues during software development.

In commercial settings (or closed source software (CSS)) the IBM Rational Jazz repository has been used in the study of software practitioners' interactions and communications largely from a SNA perspective (Nguyen, Wolf, et al., 2008; Wolf, Schroter, Damian, & Nguyen, 2009; Wolf, Schroter, Damian, Panjer, & Nguyen, 2009), offering contradictory findings to those drawn from the OSS-based body of work. Contrary to the findings reported by Shihab et al. (2009) and Shihab et al. (2010), Nguyen, Wolf, et al. (2008) revealed that about 75% of Jazz's core team members actively participated in the project's communication network. Additionally, these authors found Jazz project teams to have very inter-connected social networks, requiring few brokers to bridge communication gaps. These findings may be reflective of

software practitioners' disposition in commercial settings, where team members' motivation to contribute their knowledge is likely to be driven by greater (potentially tangible) rewards when compared to those received in OSS environments.

However, in line with the evidence reported in the OSS context (Shihab et al., 2010; Shihab et al., 2009), an earlier SNA study reported by Cataldo et al. (2006) also found that central individuals contributed the most during software development. Cataldo et al. (2008)'s study of IRC communication patterns in a large distributed system further revealed a small group of developers acting as communication hubs, and these members were also the most productive. This divergence in communication patterns noted for Jazz teams and those of other OSS and CSS projects signals the need for further confirmatory research.

Datta et al. (2011) used SNA to study the communication artefacts produced by a distributed Scrum team using the Jazz platform and found that developers tended to collaborate more as the project progressed. These authors noted that developers collaborated much more via messages than they did working on actual software development tasks. Similar evidence was revealed in a later study by Datta, Sindhgatta, & Sengupta (2012).

In another SNA study of multiple IBM Rational Jazz teams' communication, Ehrlich & Cataldo (2012) discovered that developers performed better when they occupied central positions in their specific team's communication network. However, their performance degraded when their networks were extended to multiple teams and across the entire project. Additionally, Ehrlich & Cataldo (2012) revealed that those who were parties to dense network segments demonstrated higher level of task performance. Cataldo & Ehrlich (2012) also studied IBM Rational Jazz teams' communication and revealed that teams that operated in a hierarchical communication structure completed more tasks in their iterations than those who worked in a small-world network communication structure. However, those that demonstrated the small-world communication pattern delivered higher quality software features.

Studies such as those of Bacchelli, Lanza, & Robbes (2010) and Antoniol, Ayari, Penta, Khomh, & Gueheneuc (2008) have used rather more complex techniques to analyse email and bug description information. In linking email communications to source code using regular expressions and other information retrieval approaches Bacchelli et al.

(2010) found that the analysis approach using regular expressions in emails outperformed more complex probabilistic and vector space models. Through the use of decision trees, naïve Bayes classifiers and logistic regression, Antoniol et al. (2008) were also able to classify bugs based on specific terms used in the textual descriptions of such tasks. Further, works by Baysal & Malton (2007) and Pattison, Bird, & Devanbu (2008) have discovered linkages between developers' word use and actual source code modifications.

In summarising the numerous studies just described, it is evident that a few have looked to infer the semantics of practitioners' dialogues from the text they communicated (Antoniol et al., 2008; Bacchelli et al., 2010), while many others have provided deductions based on communication frequency information (Abreu & Premraj, 2009; Bird et al., 2006a; Ehrlich & Cataldo, 2012). As noted, while text analysis methods and their associated tools, and particularly those that have been derived from psycholinguistics, have been used previously to understand and predict some aspects of software development (Junior et al., 2010), only a few studies in this domain have considered examining teams' internal behavioural processes based on their members' textual communications (Rigby & Hassan, 2007). This is in spite of the fact, as noted by Bacchelli et al. (2010), Baysal & Malton (2007) and Pattison et al. (2008), that natural language analysis techniques have proved to be effective in generating understandings of software developers' attitudes from their language processes.

Questions regarding outcome reliability and validity have also been raised for studies analysing OSS repositories, in terms of arriving at generalisable conclusions concerning software process issues. Research evidence has reported poor data quality in some repositories of OSS projects (Aune, Bachmann, Bernstein, Bird, & Devanbu, 2008; Bird et al., 2006a; Rodriguez, Herraiz, & Harrison, 2012). For instance, in their study of the Apache mailing list, Bird et al. (2006a) found it difficult to uniquely identify developers' records due to the volume of email addresses and aliases these individuals used. Further confounding issues may also be encountered when studying OSS projects because anyone is able to post messages and report bugs to their associated mailing lists, whether or not those individuals are contributing to the project (Bettenburg et al., 2007) or have a full understanding of the project.

Given these issues, coupled with the potential value of studying team interactions (noted in Section 2.4 and Section 2.5), the gaps identified above, and the growing popularity of

AGSD teams (noted in Section 2.3), it is imperative that researchers examine the contextual interactions and engagements of successful software practitioners, using representative systems, if there is to be adequate comprehension of the unique nature of these AGSD teams (Di Penta, 2012). As a first step in this regard, the following five research questions (also outlined in Chapter 1) are designed to direct the initial explorations of the collaboration patterns of successful globally distributed agile teams and to verify or challenge previous research findings (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009):

RQ1. Do communication patterns change as the software project progress?

RQ2. Is there equity in practitioners' contributions to their project?

RQ3. Are active communicators more important to their teams' collaboration?

RQ4. How are active communicators involved in task performance?

RQ5. Are practitioners' formal role assignments related to their involvement in project interactions and task performance?

Questions RQ1 – RQ 5 above are answered using quantitative analysis (including SNA) techniques (refer to Section 3.4), and are largely aimed at confirming or refuting the presence of the centralised pattern noted previously in distributed team communication networks (Cataldo & Herbsleb, 2008; Shihab et al., 2010; Shihab et al., 2009). In addition, these questions are directed to investigating the collaboration patterns of successful globally distributed agile teams and providing direction for more specific exploratory analysis regarding core developers. Additional research questions aimed at providing insights into the way core developers contribute to their teams' dynamics, which may be derived through the use of deeper contextual analysis techniques, are outlined in the following section (Section 2.7).

2.7 Wheel Structure Networks and Central Communicators

Previous research has generally established that a few individuals in a software development team dominate project communication and source code changes (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009). Evidence has also shown that, even in environments with fixed and known task assignments, specific individuals circumvent these pre-set arrangements to occupy the centre of their teams' activities (Datta et al., 2010). Such communication patterns (illustrated in Figure 2) have been studied previously in other disciplines, including

management and organizational teams and virtual research and development (R&D) groups (Ahuja, Galletta, & Carley, 2003; Guetzkow & Simon, 1955), and early works investigating the effect of this phenomenon have shown that the existence of these centralized patterns involving core group members is a positive sign for team performance (Bavelas, 1950). In similarly seminal work, Leavitt established that central individuals are vital to their teams' performance as they coordinate information flow. Central individuals are also *seen as* project leaders by others in the team, whether or not they are the formal or nominal leaders (Leavitt, 1951), and groups with central coordinators experience higher levels of group organization and task performance (in terms of speed of task completion).

While there is therefore strong interest in identifying the presence of patterns within software teams' communication and coordination practices (Cataldo & Herbsleb, 2008; Ehrlich & Cataldo, 2012; Shihab et al., 2010; Shihab et al., 2009) there has been comparatively little effort directed toward understanding why these patterns exist (Licorish & MacDonell, 2013c). In fact, evidence has revealed core communicators to be core developers (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009), an observation that should have encouraged analysis of these members' artefacts to uncover how they contribute to team dynamics and the way they are able to manage this dual presence. However, the quantitative analysis approaches that have dominated prior works considering software teams' communication artefacts are not intended to be able to reliably explain the reasons for these patterns (Di Penta, 2012; Goguen, 1993; Yu et al., 2011). Questions related to how central communicators share knowledge during their project, the initial arrangements that lead to members becoming hubs in their teams, and how the attitudes and traits these practitioners exhibit might be linked to their involvement in task performance, have not been answered. Such explorations will provide insights into the peculiarities of software team dynamics, inform appropriate team configurations, and enable the early identification of 'software gems' – exceptional practitioners in terms of both task and team performance.

Central communicators have also been previously referred to as active communicators, core communicators, core members and core developers (Crowston et al., 2006; Ehrlich & Cataldo, 2012; Rigby & Hassan, 2007; Shihab et al., 2009). In this study the terms "central communicators" and "core communicators" are used interchangeably to refer to the contributor(s) that occupy the centre of their team's communication. This pattern is

illustrated in Figure 2. The term “core developers” refers to those that are both actively involved in communication and task performance.

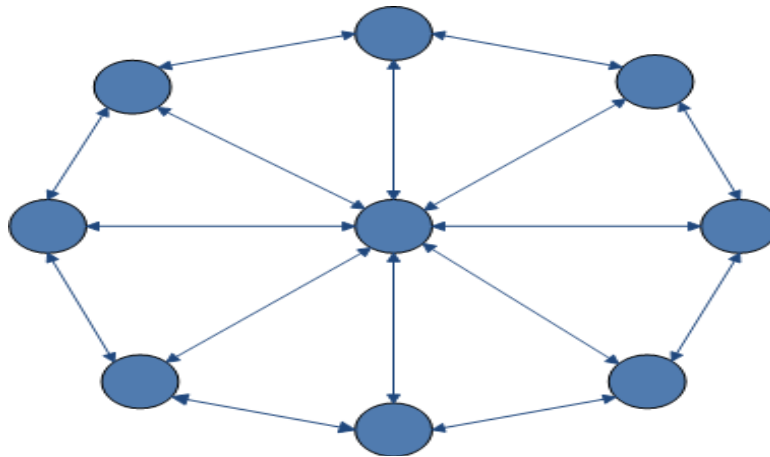


Figure 2. Abstract representation of the wheel structure communication network

This section outlines this work’s agenda that is aimed at addressing these questions through the use of deeper contextual analysis techniques (refer to Chapter 3). Firstly, summary theories regarding attitudes and team roles are provided in Section 2.7.1, directed towards supporting some specific research questions aimed at investigating core developers’ true roles in their teams. Section 2.7.2 then provides the rationale for employing a longitudinal approach for studying the changes in core developers’ attitudes, knowledge sharing behaviours and task performance over their project. Finally, Section 2.7.3 considers theories and questions aimed at investigating the *relationship between* core developers’ attitudes and knowledge sharing behaviours and their task performance.

2.7.1 Attitudes and Team Roles

As noted above, previous research has identified that an individual’s linguistic style is quite stable over periods of time, and that text analysis programs are able to accurately link language characteristics to behavioural traits (Mairesse et al., 2007; Pennebaker & King, 1999). Additionally, evidence has shown that while some team behaviours are desirable for maintaining a positive team environment, others have a negative impact on teamwork (Chang et al., 2013; Denning, 2012). Given core developers’ central position in their team communications and perceived leadership of their teams, these members’ expression of unconstructive attitudes would negatively affect their colleagues’ performance (Belbin, 2002; Benne & Sheats, 1948; Solomon, 2007). This is a particularly critical issue for agile globally distributed software development contexts, where individuals are already affected by distance and have few if any opportunities to

engage in face-to-face communication (Chang & Ehrlich, 2007; Espinosa et al., 2006) – which are shown to stimulate trust (Al-Ani et al., 2011; Krebs, Hobman, & Bordia, 2006; Zigurs, 2003). A study of core developers’ expression of attitudes is also likely to shed light on their commitment to team performance, and their effect on overall team dynamics (Allen & Meyer, 1990; Morgan & Hunt, 1994).

Similarly, an assessment of the roles core developers *actually enact* during teamwork would shed further light on their involvement in team dynamics. Roles are said to reflect the particular rights, tasks, responsibilities, expectations and behaviours that persons are expected to honour or fulfil (Bales, 1950b; Belbin, 2002). While the idea of studying and relating participants’ behaviours to roles has attracted extensive research in the psychology, sociology and management disciplines (Ashforth, 2001; Biddle & Thomas, 1966; Hellriegel & Slocum, 2007), and there has also been some consideration of this subject in software engineering (Acuna et al., 2006; Colomo-Palacios et al., 2010; Downey, 2009), this subject is rarely studied for AGSD projects (refer to Section 2.2 for further details).

Research conducted within the psychology, sociology and management disciplines has sought to inform the process of personnel assignment to jobs based on their traits and natural preferences. According to these theories, social and team role principles may be used to group individual behaviours and their personal interaction in teams, and each individual’s behavioural style is correlated with their personal preference(s) for specific roles (Belbin, 2002; Benne & Sheats, 1948; Jung, 1971). In fact, the group role concept has been consistently validated by researchers, including Bales (1950b), Benne & Sheats (1948), Margerison, McCann, & Davies (1986) and Woodcock (1989). One of the earliest and most comprehensive group role models was presented by Benne & Sheats (1948). In their study these authors identified both positive and negative group behaviours in teams. They also discovered that team social interaction is one of the main influential factors of success in group work. In total, Benne & Sheats (1948) observed 26 functional roles grouped under three dimensions of individual behaviour in teams solving problems: helpful and supportive behaviours (personal and social roles), task concerned behaviours (task roles), and debate and conflict centred behaviours (individualistic roles). Personal and social roles are said to contribute towards positive group climate, promoting encouraging, harmonising and compromising traits, while task roles are concerned with task success, contributing and initiating ideas and knowledge towards task completion. Benne & Sheats (1948) explained that individualistic roles are

more self-focused, often seeking undue recognition, and are often confrontational. Benne & Sheats (1948) noted, however, that *all* roles are important during group tasks (including individualistic roles), that the requirements for certain roles vary during different stages of teamwork, and that these roles should be adopted by various individuals at different times for group members to provide maximum contributions to the team and increase the likelihood of group success. For example, social roles may be especially necessary during times of high intensity and stressful team work – providing encouragement and support for team members – whereas task related roles may be most effective during actual task analysis and brainstorming stages. Moderate levels of individualistic roles may also be useful for maintaining high team standards through critical and constructive debates.

In considering the quite detailed model developed by Benne & Sheats (1948) it is notable that many other group role theorists have taken a slightly different perspective on this work, summarising the number of role categories provided in this early study and thereby providing more condensed models. This is evident in Bales (1950b) consideration of only task and social behaviours in his model; Woodcock (1989) considered 12 related roles; while Margerison et al. (1986) acknowledged nine. Another similar model to that of Benne & Sheats (1948) that has received considerable attention over many years for assigning team members to roles is the Belbin model (Pollock, 2009; Stevens & Henry, 1997). This model outlines nine roles for team success (Belbin, 2002), and stress the need for heterogeneity of roles during teamwork, somewhat in line with Benne & Sheats (1948)'s position on the need for all roles.

Several of the studies introduced in Section 2.6 revealed that only a small number of team members tend to dominate team communications (Cataldo & Herbsleb, 2008; Shihab et al., 2010), and that software developers' communication and coordination activities are directly related to their involvement in software tasks (Bird et al., 2006a). While numerous principles have been used to explain this pattern (e.g., Pareto principle (Shihab et al., 2010), Small-world network (Cataldo & Ehrlich, 2012; Uzzi & Spiro, 2005) and Bradman's law (Crowston et al., 2006)), previous research did not explore the actual reasons for such patterns.

Previous work has shown that these individuals occupy the centre of their teams' information sharing network and are critical to team performance (Bavelas, 1950). These members have also been shown to influence their wider teams' willingness to

adapt to change and maintain performance (Ruhnow, 2007). As noted above, however, while this pattern has been noted, questions related to the reasons for these members' extraordinary presence, and understanding the actual roles (both formal and informal) that core developers occupy in their teams, have not been answered. Such answers could provide explanations for the nature (and peculiarities) of agile globally distributed software development teams' dynamics. Knowledge and awareness of the ways in which the most active practitioners contribute their social and intellectual capital to their teams and project could help project leaders to identify exceptional software practitioners, and inform the process of assembling high performing and cohesive teams. Such findings could also inform the use of specific organizational arrangements and team configurations in support of high performers. Furthermore, the output of these explorations may lead to new requirements for collaboration and process support tools. Therefore, the following questions (briefly introduced in Section 1.5) are outlined to address these gaps:

RQ6. Do core developers' behaviours and attitudes differ from those of other software practitioners?

RQ7. What are the core developers' enacted roles in their teams, and how are these roles occupied?

2.7.2 Changes in Attitudes and Knowledge Sharing

Individuals' interactions and active involvement (denoted here by individuals' active involvement in communication networks) are major influences on knowledge creation and sharing during group work (O'Dell & Grayson, 1998; van den Hooff & de Ridder, 2004). The process of knowledge creation has been characterised as knowledge donation and knowledge collection, both of which are said to contribute to the formation of new knowledge (van den Hooff & Hendrix, 2004). In fact, it has been shown that individuals who participate by donating their intellectual capital are no more important than those who are able to get them to provide these contributions through adequate questioning (Ardichvili, Page, & Wentling, 2003). Additionally, these two activities are also posited to be influenced by cognitive and motivational factors (Hinds & Pfeffer, 2003). Cognitive factors are associated with skills and ability, whereas motivational factors are related to one's willingness to engage with other individuals in the knowledge creation and sharing process (De Vries et al., 2006).

Knowledge creation and sharing, and the variables that influence these activities, are particularly relevant to software development because of the knowledge-intensive nature of the development process. Activities centred on software development are intended to deliver a product (software) that is conceptual and intangible, with requirements that are evolving and changing, often using leading technologies and methods. As a result, dynamic knowledge sharing should be at the centre of the software development process (Baddoo, Hall, & Jagielska, 2006; Hall, Jagielska, & Baddoo, 2007). Participants' involvement in interactions and communication, and the influence of these forms of engagement on teams' performance, are the indicators used to capture the evidence that knowledge sharing is occurring in these settings. In particular, interaction, or lack thereof, has repeatedly been shown to influence the outcomes of software team processes. This phenomenon is said to be a critical success factor for software development activities (Hall, Wilson, et al., 2007), and studies assessing software practitioners' interactions have shown that team members' active involvement has a positive impact on team process. High levels of interactions have also been shown to increase individual participants' knowledge bases and enhance the likelihood of high levels of team achievement (Herbsleb et al., 2001). Furthermore, high levels of information exchange are also said to lead to improved product awareness, development task success and innovation (Damian & Zowghi, 2003; Ehrlich & Chang, 2006; Mumford & Gustafson, 1988).

While core developers (i.e., those practitioners that maintain exceptional performance in both team communications and task performance) no doubt play integral roles in these processes, it is not clear how these individuals contribute to their teams' processes over the course of their project, and how their organizational, interpersonal, intrapersonal and management competencies sustain their project's health. Additionally, there is uncertainty around what team conditions, and over which project phase(s), core developers are most engaged in their teams. Previous work has shown that practitioners' interaction patterns change over the course of a project (Cataldo & Ehrlich, 2012; Cataldo & Herbsleb, 2008; Cummings & Cross, 2003), and so longitudinal studies will uncover details that should lead to explanations for software team dynamics more fully. Evidence of how practitioners interact over the course of their project will inform targeted team strategies and phase-specific interventions. In fact, previous calls for such investigations of team dynamics have been made (Hinds & McGrath, 2006), as the

static or snapshot view does not reveal fully what actually happens over the duration of software development projects.

The utility of a longitudinal approach in studying software teams has also been demonstrated previously. For instance, Rowley & Lange (2007) applied the Tuckman (1965) model of team development to study agile teams' evolution² and found the forming, storming, norming and performing stages were somewhat cyclic, and specific techniques and approaches were applicable to multiple stages of team development, as against a linear team evolution. Ruhnow (2007) found that once core development team members embraced specific tools and techniques during a software project it was easy to get the extended team on board to use the same tools and techniques. This finding endorses the viewpoint that understanding these core members could potentially bring value to their wider teams, and provide insights relevant to overall project governance.

As noted previously, linguistic studies have shown that while individual language use is stable over time, the way individuals communicate is also influenced by their context and local settings (Mehl & Pennebaker, 2003). In software development settings, negative and cynical team behaviours can have a negative impact on team harmony and cohesion (Chang et al., 2013). This will in turn negatively affect team performance (Espinosa, Slaughter, Kraut, & Herbsleb, 2007). The opposite is likely to occur in more optimistic environments where teams share a single vision. Studies considering the effect of group norms on individual willingness to share their intellectual capital have indeed supported this reasoning (van den Hoof, de Ridder, & Aukema, 2004). Given that core developers occupy the centre of their teams' communication, are seen as project leaders (whether or not they are assigned to formal leadership roles (Hinds & McGrath, 2006)), and that they coordinate information flow and knowledge sharing (Leavitt, 1951), an understanding of core developers' attitudes and knowledge sharing behaviours will be useful in informing strategies aimed at maintaining an optimistic and positive team climate, and ultimately, positive team performance.

² These authors studied evolution by considering changes in various aspects of software teams' processes over seven iterations (Rowley & Lange, 2007). This operationalization of evolution does not necessarily reflect evolution in the true sense, which, according to the Oxford English Dictionary is "the process by which different kinds of living organism are believed to have developed from earlier forms during the history of the earth" (OED-Online). Thus, this work uses a similar approach to (Rowley & Lange, 2007), but considers this approach to be the *changes in core developers processes over time*, as against their evolution.

While changes in core developers' activities have been studied, this subject has been approached only from a quantitative perspective – typically involving numbers of code commits and messages exchanged (Cataldo & Herbsleb, 2008; Robles, Gonzalez-Barahona, & Herraiz, 2009). As noted above, there is now widespread recognition that supplementing quantitative analyses with more exploratory investigations offers avenues for outcome triangulation as well as the provision of additional insights into the software development process (Di Penta, 2012; Easterbrook et al., 2008). In particular, the early theory of SNA (which is often used for assessing teams' interactions) was only recommended for *estimating* interpersonal relationships during group work (Moreno, 1953). Erlin, Yusof, & Rahman (2008) and Jamali & Abolhassani (2006) highlighted that while SNA theories indeed provide useful tools for assessing some aspects of social structures, such as measures of centrality, cliques or sub-networks and density using visualisations and mathematical analysis, such measures may not be so useful in explaining the reasons for or consequences of social structures.

Revealing the process of how developers become core (through the use of deeper data analysis approaches) could help project leaders to identify and encourage software gems very early in their project. Some developers may occupy natural roles, such that, regardless of the project environment, these individuals may function in a certain way based on their natural preferences (Belbin, 2002). On the other hand, others may emerge into specific roles given their teams' demands and/or their specific task assignments (Hackman, 1986; Hoda, Noble, & Marshall, 2010a). Knowledge and awareness of these different developers and the way they work will help project leaders to identify software development leaders early, and in assembling high performing and cohesive teams. Accordingly, the following questions are provided (also briefly presented in Section 1.5) to study this issue:

RQ8. Do core developers' attitudes change as their project progresses?

RQ9. How do core developers share knowledge over the course of their project?

RQ10. What initial team arrangements lead to developers becoming hubs in their teams?

2.7.3 Attitudes, Knowledge Sharing and Task Performance

Software engineering research examining teams' communication has focused primarily on the use of social network related measures, and particularly measures related to

centrality and closeness (Bird et al., 2006a; Cataldo & Ehrlich, 2012; Datta et al., 2010; Ehrlich & Cataldo, 2012; Hinds & McGrath, 2006; Zhou & Mockus, 2011). In fact, as noted in Section 2.7.2, the studies that have concluded that just a few individuals contribute the most to communication and task performance have generally used frequency-based analysis techniques (Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009), and while there have been some efforts to understand the characteristics of core developers (Cataldo & Herbsleb, 2008; Robles et al., 2009), these works did not probe the reasons underlying the ‘core developer’ phenomenon. While frequency-based analysis techniques do enable the detection of certain patterns, and so provide a partial understanding of software teams’ behavioural processes, there are limitations on the effectiveness of these approaches in informing our understanding of the deeper psychosocial nature of team dynamics (Di Penta, 2012).

Knowledge sharing studies have shown that the willingness of individuals and teams to actively participate in knowledge sharing and contribute to team performance is linked to multiple factors (Hinds & Pfeffer, 2003). For instance, knowledge sharing has been linked to social motivation (e.g., trust (Inkpen & Tsang, 2005; Levin & Cross, 2004)), rewards and incentives (Kalman, Monge, Fulk, & Heino, 2002), cognitive factors (De Vries et al., 2006), and other organisational reasons (Szulanski, 2000). While there is some uncertainty around the effects of incentives and rewards on individuals’ active participation in knowledge sharing (Bock & Kim, 2002), social motivation theory has proved to be generally effective for predicting participation in knowledge sharing (Geen, 1991; Inkpen & Tsang, 2005; Levin & Cross, 2004). According to social motivation theory, teams’ interpersonal interactions and norms have an impact on individual members’ motivation to perform (Geen, 1991). Thus, certain supportive behavioural norms at the team level are likely to encourage individual performance (Quigley, Tesluk, Locke, & Bartol, 2007). This position may be especially valid given that knowledge sharing is a social process (Bock & Kim, 2002).

Thus, core developers’ engagements in their teams’ knowledge sharing process and task performance over their project may be linked to specific events and task arrangements (Hackman, 1992). Particularly, when core developers operate during periods of positive and social behavioural climate, these individuals may be most happy to perform (De Vries et al., 2006; Zakaria et al., 2004). Similarly, in a more cognitive and evaluative environment, core developers may exhibit eagerness to perform. Studying these members’ attitudes and knowledge sharing behaviours, and the way these are linked to

their involvement in their teams' task performance, would shed further light on agile globally distributed teams' dynamics.

Although it is understood that core developers are invaluable to their teams, beyond simply liaison and task change roles (Cataldo & Herbsleb, 2008), there still remain doubts regarding when these individuals are more or less likely to contribute the most to task performance and when their teams are most likely to benefit from their knowledge and experiences. These insights would be useful for understanding the specific traits of less prudent team members that are likely to complement these core individuals. Additionally, such understandings would inform specific project arrangements that can enhance the satisfaction of core developers. Further, such answers would reveal how software teams should be staffed during core developers' less productive periods. Answers to these questions will provide valuable insights for software project governance. Psycholinguistics and directed content analysis techniques provided this work an avenue to answer the following questions (also briefly introduced in Section 1.5):

RQ11. How do core developers contribute to task performance over their project?

RQ12. Are core developers' contributions to task performance linked to their attitudes?

RQ13. Are core developers' contributions to task performance linked to their contribution of knowledge?

2.8 Chapter Summary

This chapter has provided a survey of relevant theories in support of this research project. Given on-going evidence of inadequately performing software teams, and particularly after continuous interventions related to software methodologies and tools, there has been a recent shift in focus towards understanding the human processes that are involved during software development. This move is fitting given that most of the risk issues revealed in the literature may be deemed people driven. This work also studies the human issues involved during software development through the use of techniques that are grounded in the psychology and social science paradigms. To provide grounding for the work that is performed in this study, theories supporting team composition, psychology and software engineering human factors frameworks were

reviewed. The AGSD concept was also introduced, and it was shown that agile techniques conflict with the realities of globally distributed software development.

Given the reliance on communication technologies in AGSD environments, and textual communication in general, the study of communication was considered, along with approaches that consider the way attitudes are revealed in text. It was also observed that studies that have examined software developers' artefacts have provided multiple insights through such means. However, while this form of communication is widely investigated, studies examining software practitioners' artefacts have largely used quantitative and frequency-based analysis approaches. This represents a limitation to the level of insights that is provided by these works, and particularly for the reason for the centralised pattern that is noted for software teams' communication networks.

Previous works have shown that core communicators are vital to their teams' performance as they coordinate information flow, and these members are perceived as project leaders. Thus, this work explores the collaboration patterns of successful globally distributed agile teams to verify or challenge previous research findings. This first step provides the platform for further in-depth examinations aimed at providing insights into the actual role of core developers, and the way these members' attitudes and knowledge sharing behaviours change over the course of their project. Further, this work also considers how core developers' expressions of attitudes and their involvement in knowledge sharing are linked to their task performance. These issues are investigated through the research questions that were outlined in this chapter.

Now that the research gaps have been identified and the research questions specified, attention moves to the research methodology. The following chapter (Chapter 3) outlines the research methodology and design, including the techniques that are used for data analysis and operationalization of the constructs introduced in the 13 research questions.

Chapter 3. Research Methodology and Design

Studies in the SE discipline examining human processes through the analysis of repository data have regularly employed frequency-based approaches (e.g., SNA) (Bird et al., 2006a; Cataldo et al., 2006; Herbsleb et al., 2000). Such approaches are in line with the way early SE studies frequently considered only the technical aspects of this activity (Easterbrook et al., 2008; Glass, Vessey, & Ramesh, 2002). However, it is now generally understood, and there is growing recognition in SE, that studying technical aspects of the software process in isolation may present a limitation to the evidence that such projects can provide (Johnson & Onwuegbuzie, 2004).

This view is supported by researchers in the Information Systems (IS) discipline, where it has been repeatedly shown that there are sound reasons for studying deeper organisational and behavioural issues (Klein & Myers, 1999; Ramesh, Glass, & Vessey, 2004), apart from the technical facets of IS. What is more, it has been shown that there is *no other way* to undertake certain types of enquiry (such as to provide explanations for the details of human and organisation processes and how these may be harnessed to deliver maximum benefit for the software development community) than to engage the more conventional research approaches as have been used and tested in other disciplines (Klein & Myers, 1999; Vessey, Ramesh, & Glass, 2002). In fact, Vessey et al. (2002)'s comprehensive survey of the IS research domain outlined that success in studying some SE and IS issues that are intended to promote understanding of individual and organisational phenomena may not be achieved without adequate understandings (and expertise) of the techniques applied in the social and behavioural sciences, management and psychology domains.

With grounding in these disciplines, this work utilises suitable methodologies to study globally distributed agile team dynamics and, in particular, the attitudes, roles and knowledge sharing behaviours of core developers. To this end, this chapter outlines how appropriate research tools and techniques are selected in order to make the research outputs and findings of this study relevant for SE theory and practice. The approaches selected also ensure that the findings of this work are applicable to the issues under consideration. As a preliminary step in this direction, this chapter firstly introduces the way research methods are selected with an emphasis on SE research (Section 3.1). Section 3.2 considers the primary research dichotomy (positivist and interpretivist), and how these approaches influence the way research questions are formed and methods are

selected, towards highlighting how studies are linked to research processes and paradigms. This step informs the next phase – the selection of an approach for guiding the work undertaken in this project, which is subsequently introduced in Section 3.3. The Case Study method – the approach chosen in this study – is then introduced in Section 3.4. The subsequent section (Section 3.5) describes the process of theorising in SE, with particular emphasis on the path chosen during the provision of conjectures that may form the basis of explanation theories that this work provides. Finally, Section 3.6 presents a summary of the details and discussions provided throughout this chapter.

3.1 Selecting a Research Method

Selecting appropriate methods with which to conduct empirical SE research often poses many challenges for researchers (Easterbrook et al., 2008; Lázaro & Marcos, 2006). This may generally represent a lack of knowledge about the range of techniques available and how these may be used. A simple way for researchers to address this activity is to consider the phenomenon of interest (Lázaro & Marcos, 2006; Leech & Onwuegbuzie, 2009). However, this issue is sometimes confounded by the philosophical position (the way truth is perceived) of those undertaking research, which also tends to influence the approaches that are selected for conducting research (Galliers & Land, 1987).

The research questions (or the phenomena of interest) are often driven by the prevalence or lack of theory in the specific domain (Newman & Benz, 1998). Typically, when there are mature theories in the domain under consideration research questions are generally aimed at verifying, testing and modifying these theories (e.g., Does the absence of testers cause software project failure? Does project management tool use delay software project delivery? and so on). In such instances the researchers' main intent is to check for relationships, and thus, their studies generally employ a positivist and quantitative approach (Easterbrook et al., 2008; Lázaro & Marcos, 2006). On the other hand, where there are little or no theories available, research questions are largely exploratory, aimed at theory initiation or building (e.g., What are the actual tasks of software testers during software development? How do the duties of software testers differ from those of programmers during software development? and so on). Accordingly, these works employ more constructivist and often qualitative approaches (Easterbrook et al., 2008; Lázaro & Marcos, 2006).

As noted earlier, researchers' beliefs (see further discussions in Section 3.2) also impact the way they approach the research process (Leech & Onwuegbuzie, 2009). It has also been highlighted that practical considerations of time, budget and access to data may influence the study approach that is finally adopted by researchers (Easterbrook et al., 2008). Access to data is particularly troublesome, especially when researchers aim to assess what participants actually do, against what is self-reported. Another issue that challenges researchers is the quality of data available for studying human issues. All of these issues interplay during the consideration of appropriate methods.

Of the issues highlighted above, a common determinant used for selecting one research approach over another is the way truth is perceived by the researcher (Easterbrook et al., 2008; Onwuegbuzie & Leech, 2005). In fact, early theorists have even argued that research should adopt an exclusive approach (either quantitative or qualitative) (Howe, 1988). However, in recent times it has become quite common for research to employ multiple techniques (Fitzgerald & Howcroft, 1998). These issues are considered in the next section (Section 3.2), and inform the selection of the methodology adopted in this work.

3.2 Research Perspectives - Positivist versus Interpretivist

As just outlined, the philosophical perspective of researchers often drives the way truth is perceived. Such perceptions are influenced by the distinctions made between ontology and epistemology (Creswell, 2002). The ontological perspective considers the nature of the world or reality, whereas, the epistemological stance explains the meaning of human knowledge and how it is obtained (Bryman, 1984). To the positivist (also called reductionist or purist) knowledge exists independent of individuals, and objective and verifiable procedures may be used to help individuals understand parts of knowledge, which may allow inferences to be made towards understanding the whole (Onwuegbuzie & Leech, 2005). On the other hand, the constructivist (also called interpretivist or situationalist), opposes the view that reality exists independent of individuals, and so advance the position that understanding of the social world is specific to the frames of reference of its examination (Klein & Myers, 1999). These two positions (positivist and constructivist) shape the way methodologies are adopted by researchers (refer to Table 1 for details – taken from Fitzgerald & Howcroft (1998)). Positivists normally prefer to frame research around verifiable hypotheses or research questions, and in the process, they tend to use methods such as experiments, surveys

and case studies that are quantifiable in nature (Hussey & Hussey, 1997). In contrast, constructivists focus on how and why individuals make sense of the world, preferring to be more exploratory in nature. Accordingly, researchers employing this approach typically use methods such as exploratory case studies, ethnography, grounded theory or anthropology to provide richer accounts of the phenomena (Klein & Myers, 1999) (refer to Table 1 for details).

Both of these approaches possess strengths and weaknesses (Onwuegbuzie & Leech, 2005). While a theory-driven approach utilised for studies employing a positivist focus may provide strength in the way findings are analysed (in relation to theories), deciding on variables prior to the advanced stages of the research may result in researchers ignoring important issues, which may limit the accuracy and applicability of the study contributions (Easterbrook et al., 2008). Likewise, employing qualitative methodologies provides its own challenges related to subjectivity and generalisability, especially in the way multiple contradictory findings are reported in the study of a single phenomenon (Onwuegbuzie & Leech, 2005) and the fact that data from qualitative studies and the procedures employed during data interpretation are rarely made public (Constas, 1992, p. 254).

Hence, strict adherence to a particular paradigm may result in limitations to the research contributions. For that reason, it is often recommended that researchers should aim to reduce the weaknesses inherent in both paradigms, while exploiting their strengths – by employing a pragmatic approach (Leech & Onwuegbuzie, 2009; Onwuegbuzie & Leech, 2005; Schultz & Hatch, 1996). Such an approach to research is generally recommended for overcoming the shortcomings of individual techniques, and unearthing deeper insights which may lead to more complete understandings of the issues under consideration (Johnson & Onwuegbuzie, 2004; Leech & Onwuegbuzie, 2009). This approach involves collecting, analysing and interpreting both quantitative and qualitative data during the investigation of a single observable ‘fact’ (Lázaro & Marcos, 2006; Leech & Onwuegbuzie, 2009).

The move to combine approaches is in line with the viewpoint that conforming to a specific epistemological perspective may delay the delivery of meaningful observations, and that research methodologies represent tools and techniques that are used to help with knowledge discovery through systematic and coherent enquiry (Fitzgerald &

Howcroft, 1998; Onwuegbuzie & Leech, 2005). This stance is also adopted in this work, the details of which are discussed in the next section (Section 3.3).

Table 1. Constructivist versus Positivist research dichotomy

PARADIGM LEVEL	
Constructivist	Positivist
No universal truth. Understand and interpret from researcher's own frame of reference. Uncommitted neutrality impossible. Realism of context important.	Belief that world conforms to fixed laws of causation. Complexity can be tackled by reductionism. Emphasis on objectivity, measurement and repeatability.
ONTOLOGICAL LEVEL	
Relativist	Realist
Belief that multiple realities exist as subjective constructions of the mind. Socially-transmitted terms direct how reality is perceived and this will vary across different languages and cultures.	Belief that external world consists of pre-existing hard, tangible structures which exist independently of an individual's cognition.
EPISTEMOLOGICAL LEVEL	
Subjectivist	Objectivist
Distinction between the researcher and research situation is collapsed. Research findings emerge from the interaction between researcher and research situation, and the values and beliefs of the researcher are central mediators.	Both possible and essential that the researcher remain detached from the research situation. Neutral observation of reality must take place in the absence of any contaminating values or biases on the part of the researcher.
METHODOLOGICAL LEVEL	
Qualitative	Quantitative
Determining what things exist rather than how many there are. Thick description. Less structured and more responsive to needs and nature of research situation.	Use of mathematical and statistical techniques to identify facts and causal relationships. Samples can be larger and more representative. Results can be generalised to larger populations within known limits of error.
Exploratory	Confirmatory
Concerned with discovering patterns in research data, and to explain/understand them. Lays basic descriptive foundation. May lead to generation of hypotheses.	Concerned with hypothesis testing and theory verification. Tends to follow positivist, quantitative modes of research.
Induction	Deduction
Begins with specific instances which are used to arrive at overall generalisations which can be expected on the balance of probability. New evidence may cause conclusions to be revised. Criticised by many philosophers of science, but plays an important role in theory/hypothesis conception.	Uses general results to ascribe properties to specific instances. An argument is valid if it is impossible for the conclusions to be false if the premises are true. Associated with theory verification/falsification and hypothesis testing.
Field	Laboratory
Emphasis on realism of context in natural situation, but precision in control of variables and behaviour measurement cannot be achieved.	Precise measurement and control of variables, but at expense of naturalness of situation, since real-world intensity and variation may not be achievable.

3.3 A Pragmatic Research Approach

This research adopts a pragmatic approach (Newman & Benz, 1998), where the research problems under consideration drive the methods and techniques that are selected for this project. Given the range of issues outlined in Chapter 2, conforming to a single philosophical position, positivist or constructionist, would not be ideal for the work conducted in this study (Onwuegbuzie & Teddlie, 2003). The research questions derived in Chapter 2, and noted below in Figure 4, are largely exploratory in nature (e.g., RQ6. Do core developers' behaviours and attitudes differ from those of other

software practitioners?, RQ7. What are the core developers' enacted roles in their teams, and how are these roles occupied?), which is influenced by the state of research for the phenomena under observation (refer to Section 3.2 for examples of the way such research issues are addressed). While a substantial body of research has examined software teams' communication artefacts (Aranda & Venolia, 2009; Bird et al., 2006b; Cataldo & Ehrlich, 2012; Crowston et al., 2008; Datta et al., 2010; Datta et al., 2011; Ehrlich & Cataldo, 2012; Nguyen, Wolf, et al., 2008; Shihab et al., 2009), and the centralised communication pattern has been observed for most software teams (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009), previous work has not investigated the reason(s) for this phenomenon. Previous research has shown that these central individuals occupy the core of their teams' information sharing network and are critical to team performance (Bavelas, 1950). These members have also been shown to influence their wider teams' willingness to adapt to change and maintain performance (Ruhnnow, 2007). Thus, RQ6 and RQ7 (refer to Section 2.7.1) investigate *the reasons* for core developers' extraordinary presence, and provide understanding for the actual roles (both formal and informal) core developers occupy in their teams. This aspect of the research is aimed principally at theory initiation and building, and so employs more qualitative approaches to address these objectives. This approach is similarly adopted for the research questions in Section 2.7.2 and Section 2.7.3.

On the other hand, other aspects of the research are intended to verify or refute previous quantitative research findings, such as those of Bird et al. (2006a), Cataldo & Herbsleb (2008), Cataldo et al. (2006) and Shihab et al. (2009) (e.g., RQ1. Do communication patterns change as the software project progresses?, RQ2. Is there equity in practitioners' contributions to their project?) (refer to Section 2.6), and in the process provide confirmation or otherwise for these theories. This demands the utilisation of techniques associated with a quantitative framework, and thus, such approaches are also adopted in this work.

Given the data intensive nature of the work, quantitative measures are also used for data reduction, data cleaning and the analysis of large samples of numeric data in the early part of the work (Lázaro & Marcos, 2006; Onwuegbuzie & Leech, 2005) – in the process providing confirmation for previous evidence reported (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009) and preliminary extensions of previous theories. The more qualitative aspects of the work are guided by

thematic analysis techniques, towards the provision of initial theories (Onwuegbuzie, 2003) – in consideration of the issues that have been overlooked in previous work. Quantitative measures are then used to identify meta-themes and relationships among themes revealed through both qualitative and quantitative observations (as per the work of Barcellini, Detienne, Burkhardt, & Sack (2008), for example).

These approaches are utilised together to provide multiple strengths to the work under consideration. In using quantitative techniques to analyse themes revealed from qualitative data analysis, this study provides deeper levels of interpretation for the exploratory aspects of the work. Additionally, qualitative aspects of the work help to explain statistically significant findings discovered during the quantitative elements of the work, and also act as a means of providing triangulation for the techniques selected. These approaches, as utilised under the principles of the case study method, are outlined in the next section (Section 3.4).

3.4 Case Study Method and Study Design

In light of the research issues and questions presented in Chapter 2 (see consolidated research questions in Figure 4), and the subsequent discussions provided in Section 3.1, Section 3.2, and Section 3.3 relating to the way research is conducted in SE and the rationale for using the pragmatic approach selected in this work, this study utilises a mixed method approach (Tashakkori & Teddlie, 1998). This approach is implemented under a case study design, with the aim of contributing confirmations and initial theories for explaining (Gregor, 2006) the nature of agile globally distributed software teams' dynamics and the true role of core developers, the way core developers' attitudes and knowledge sharing behaviours change over their project, and the relationship between core developers' attitudes and knowledge sharing behaviours and their involvement in task performance (see the discussion on the process of theorising in Section 3.5). Findings from these enquiries provide contributions to software engineering theory and practice.

According to Yin (2003), case studies are used to investigate contemporary issues in real settings. In particular, Yin posited that this method is generally suitable when there are unclear boundaries between phenomena and context (Yin, 2003). Thus, this method provides an avenue to understand how, when, and why events occur (Flyvbjerg, 2006), in line with the objectives of this study. Exploratory aspects of the case study are used

to provide initial theories, while previous theories are assessed using more confirmatory methods (Easterbrook et al., 2008).

A case study may employ purposive sampling in order that relevant cases are selected for observation (Yin, 2003). Sometimes the most representative cases are selected, but abnormal cases may also provide interesting observations (Flyvbjerg, 2006). Such a mixed approach is used during this work's case selection process, and deliberate efforts are employed to ensure that interesting variations in the repository are captured during data sampling (refer to Section 3.4.2 for further details). Additionally, while it is not unusual for a case study to be conducted using one case, research employing multiple cases provides stronger claims for validity (Easterbrook et al., 2008). While one large case organisation is used during this work (IBM Rational Jazz – see discussions in Section 3.4.1), multiple teams are investigated as individual cases, and an embedded case approach is also used to study individual practitioners in each team (refer to Figure 3 for illustration).

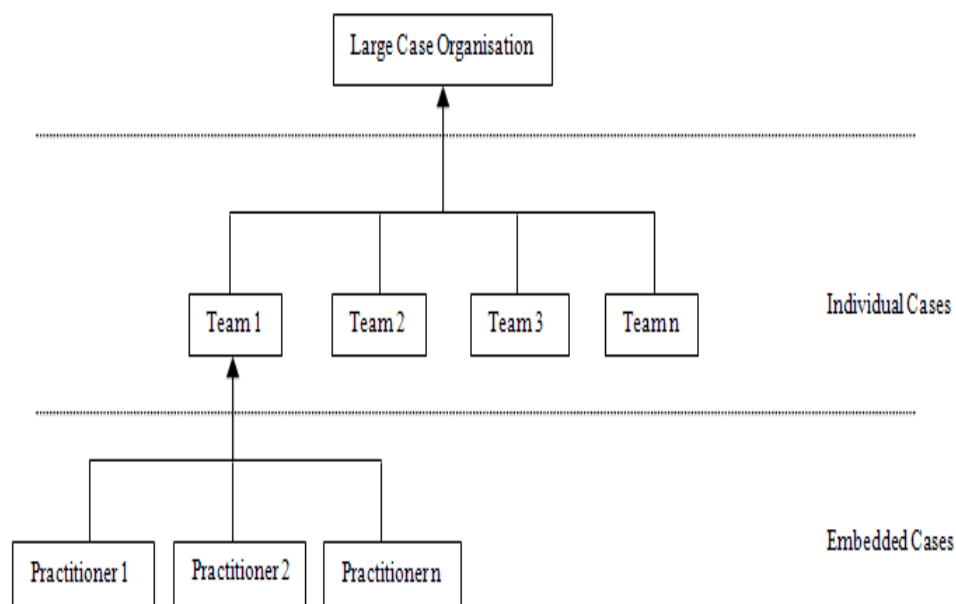


Figure 3. Case study model

Depending on the type of case study methodology that is adopted (whether positivist or constructivist) appropriate data collection and analysis techniques are utilised, where positivist studies use mostly quantitative techniques (Yin, 2003), while constructivists use qualitative data and associated techniques (Walsham, 1993). As noted in Section 3.3, this work utilises both approaches (quantitative and qualitative) to fulfil the study aims and objectives. The unit of analysis provides the basis for how data is collected and analysed, whether at the company level, project level, team level or individual level

(Easterbrook et al., 2008). This study utilises multiple units of analysis, at the company level, team level and individual level (Runeson & Host, 2009). Thus, the analysis and findings in this work are provided accordingly – firstly at the level of the team and organisation, and then at the level of the individual. At times discussions are also provided at the individual level and then extrapolated to the team and organisation, and over project phases.

Research in SE and IS has utilised the case study method to investigate a range of issues (and many of these studies have been described in the previous chapter). Bird, Nagappan, Devanbu, Gall, & Murphy (2009) employed the case study method to investigate the impact of distributed development on software quality. The single case design was also employed by Nagappan, Murphy, & Basili (2008) to observe the effect of organisation structure on software quality. A longitudinal multi-case design was employed by McGann & Lyytinen (2008) in the examination of the way improvising affects software evolution. Gaye, Butler, & Finnegan (2010) utilised the case study method to study coordination mechanisms in a global software team at a Fortune 100 telecommunication company, using a single case design. Finally, the single case design was also utilised by Moe, Dingsoyr, & Dyba (2010) to investigate teamwork challenges in self-organising teams.

The approaches implemented by studies such as McGann & Lyytinen (2008) and Nagappan et al. (2008) and the guidelines outlined by Yin (2003) and Runeson & Host (2009) provide foundation for the work conducted in this study, and inform the way this research project is designed along with the techniques that are selected for collecting, analysing and interpreting the data representing specific properties of the population under consideration.

The study is conducted using a multi-phase approach, using both confirmatory and exploratory analysis techniques, where each phase builds on the other (refer to Figure 4 for consolidated research questions under each research phase). Since this study uses archival data, data mining principles are used for data collection, pre-processing and preliminary data exploration (refer to Section 3.4.2). Extracted data are then further explored and analysed using SNA (refer to Section 3.4.3). These activities are conducted in the first research phase, for confirmatory analysis (Phase 1), to provide insights into the collaboration patterns of successful globally distributed agile teams (in answering RQ1 – RQ 5, refer to Figure 4).

Linguistic Analysis (refer to Section 3.4.4), Content Analysis (refer to Section 3.4.5), and statistical analysis techniques are then applied to the pre-processed data in two further rounds of exploratory analyses. Static/project snapshot analyses are first conducted in a second research phase (Phase 2) to provide new insights into the true role of core developers. This undertaking is aimed at answering the second set of research questions (RQ6 and RQ7) in Figure 4 to provide explanations from which initial explanatory theories could be generated (refer to Section 3.5 for details). Longitudinal analyses are then conducted in a third research phase (Phase 3) to provide further understandings of the changes in core developers' attitudes, knowledge sharing and task performance and the relationship between these variables. This phase of analyses is aimed at answering the final set of research questions (RQ8 – RQ13) in Figure 4 towards extending the insights from the second phase (Phase 2).

The following sections provide a description of the data repository that is extracted and explored during this work (Section 3.4.1), and elaborations of the techniques and procedures utilised to fulfil the study's agenda (see Section 3.4.2, Section 3.4.3, Section 3.4.4 and Section 3.4.5, respectively).

<p>Exploring the collaboration patterns of successful globally distributed agile teams (Phase 1)</p> <p>RQ1. Do communication patterns change as the software project progresses?</p> <p>RQ2. Is there equity in practitioners' contributions to their project?</p> <p>RQ3. Are active communicators more important to their teams' collaboration?</p> <p>RQ4. How are active communicators involved in task performance?</p> <p>RQ5. Are practitioners' formal role assignments related to their involvement in project interactions and task performance?</p>
<p>The true role of core developers (Phase 2)</p> <p>RQ6. Do core developers' behaviours and attitudes differ from those of other software practitioners?</p> <p>RQ7. What are the core developers' enacted roles in their teams, and how are these roles occupied?</p>
<p>Changes in core developers' attitudes, knowledge sharing and task performance (Phase 3 (a))</p> <p>RQ8. Do core developers' attitudes change as their project progresses?</p> <p>RQ9. How do core developers share knowledge over the course of their project?</p> <p>RQ10. What initial team arrangements lead to developers becoming hubs in their teams?</p> <p>The relationship between core developers' attitudes and knowledge sharing behaviours and task performance (Phase 3 (b))</p> <p>RQ11. How do core developers contribute to task performance over their project?</p> <p>RQ12. Are core developers' contributions to task performance linked to their attitudes?</p> <p>RQ13. Are core developers' contributions to task performance linked to their contribution of knowledge?</p>

Figure 4. Consolidated research questions

3.4.1 Study Repository

The repository that is selected for examination in this work is called IBM Rational Jazz (based on the IBM^R Rational^R Team ConcertTM (RTC)³). Jazz, created by IBM, is a fully functional environment for developing software and managing the entire software development process, including project management, project communication and coding (Frost, 2007) (see a breakdown of the components of the Jazz platform in Figure 5, see <http://www.jazz.net> for further details). In fact, collaboration and awareness support is the premise on which Jazz was built, where IBM's idea was to integrate all aspects of software development in one toolset (Herzig & Zeller, 2009) and provide unhindered project awareness for team members. In this regard, this software includes features for work planning, software builds, code analysis, bug tracking and version control functionalities in one system (Rich, 2010). Traceability for these different features is provided by the tool's reporting functionalities. Changes to source code in the Jazz environment are only allowed as a consequence of earlier tasks created, such as a defect, a task or an enhancement request. Features and artefacts are tracked using work items (WIs), and a WI represents a single task⁴ classified as one of the aforementioned. Defects are tasks related to bug fixing, design documents, documentation or support for the RTC online community are labelled as tasks, while enhancements are related to new functionality or the extension of system features (Ehrlich & Cataldo, 2012). Team member communication and interaction around WIs are captured by Jazz's comment or message functionality. During development at IBM, project communication, the content explored in this study, was actually enforced through the use of Jazz itself (Nguyen, Wolf, et al., 2008).

IBM has afforded this work an opportunity to study an instance of the Jazz repository (via the IBM Academic Initiative) comprising a large amount of software development process data from planning, development and management activities across the United States, Canada and Europe. This release includes teams' artefacts that were created during the development of the now commercially available RTC. In Jazz, specific teams are responsible for various aspect(s) or component(s) of the overall Jazz project (e.g., Jazz Community Portal or Jazz Visual Studio Client). It is also not uncommon for team members to work across many teams occupying different roles (see an illustration of the

³ IBM, the IBM logo, ibm.com, and Rational are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

⁴ The terms "task" or "WI" are used interchangeably during this work to refer to a software feature.

way artefacts and teams' members are arranged in the Jazz repository in Figure 6). Each team has multiple individual roles, with a project leader responsible for the management and coordination of the activities undertaken by the team (Ehrlich & Cataldo, 2012). Project leaders report progress to a project management committee (PMC), which formulates and oversees the project goals.

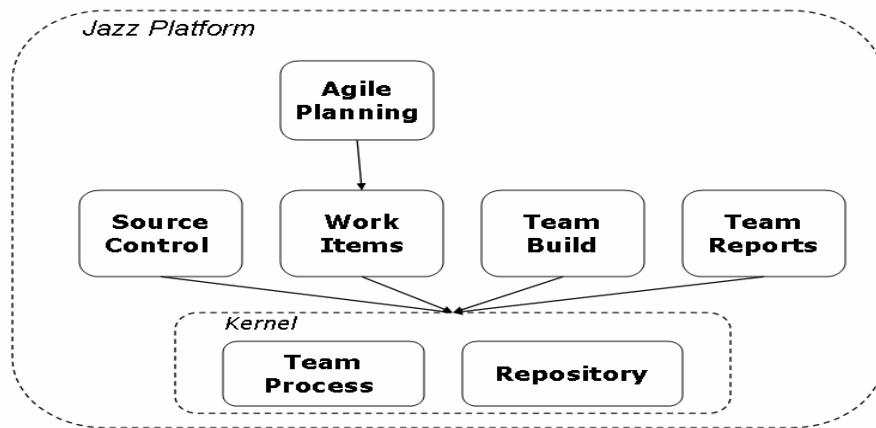


Figure 5. Components of the Jazz platform

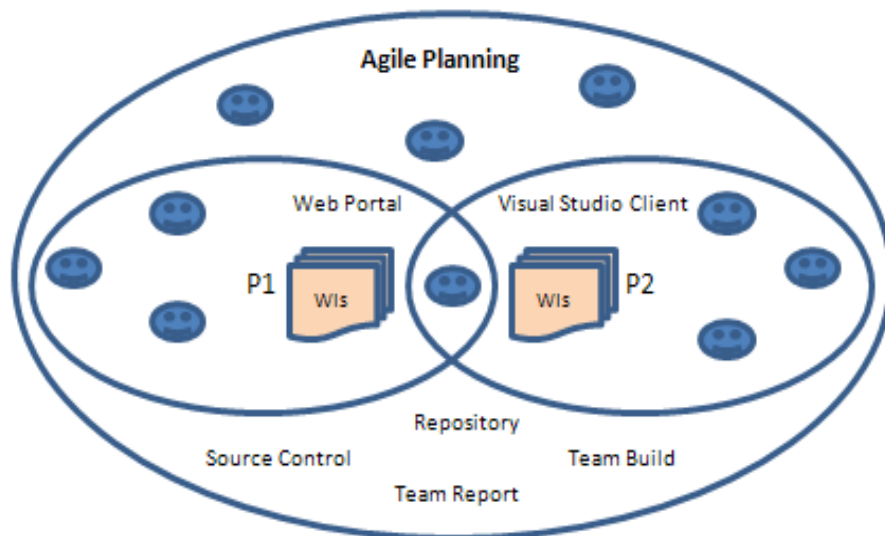


Figure 6. Teams' arrangement in Jazz

Jazz teams use the Eclipse Way (agile-based) methodology for guiding the software development process (Frost, 2007). This methodology outlines iteration cycles that are six to eight weeks in duration, comprising planning, development and stabilising phases. IBM Rational software engineers note that agile practices form the core of the Eclipse Way methodology, and that the agile tenets “iterative, reflect, adapt, incremental, feedback” were central to the way they developed the RTC (refer to <http://www.ibm.com/developerworks/rational/library/edge/08/jul08/vanVelzen/> for a relevant article by Jazz Solution Architect Ton van Velzen). During Jazz development,

builds are executed after project iterations; also called project milestones. All information regarding the software process (project management, tracking and planning, project coordination and communication and software building (coding)) is stored in a server repository, which is accessible through a web-based or Eclipse-based (RTC) client interface (Wolf, Schroter, Damian, Panjer, et al., 2009).

While the criteria for software project success generally relate to projects being completed on time, on budget and with the required features and functionality (Standish Group, 2009), others assert that measures related to software projects' impact on the development organization, post-release customers' reviews, and actual software usage are also relevant project success indicators (Espinosa et al., 2006). Accordingly, given the impact IBM Rational products (included in the Jazz repository) have had on IBM and many other organizations, with over 30,000 companies using these tools, and that these products have been positively reviewed and tested by those companies, it is contended here that Jazz teams are successful (see <http://www.jazz.net> for details). Thus, this study provides reflections of *successful* agile globally distributed software team dynamics.

Beyond Jazz (Nguyen, Wolf, et al., 2008; Treude, 2010), many tool repositories have been used previously in the observation of software processes and metrics (Bachmann & Bernstein, 2009; Bird et al., 2006a; Dutoit & Bruegge, 1998; Edwards, Puckett, & Jolly, 2006; Zimmermann & Nagappan, 2008). These studies have mostly employed OSS repositories in their enquiries (Bachmann & Bernstein, 2009), or have provided evidence derived largely from analytical and mathematical procedures, particularly SNA (Datta et al., 2010; Nguyen, Wolf, et al., 2008) and other frequency-based measures (Shihab et al., 2010). For instance, Zimmermann & Nagappan (2008) used SNA to identify defect prone binaries, Bird et al. (2006a) employed SNA to study the developers' communication and coordination activities, Datta et al. (2011) also employed SNA to study the evolution of developers' collaboration, and finally, Shihab et al. (2010) used frequency-based techniques to study the communication patterns of OSS developers. Among the works examining software artefacts, studies have used repositories of projects such as Apache (Bird et al., 2006a), Eclipse (Abreu & Premraj, 2009), GNOME (Bachmann & Bernstein, 2009), Netbeans (Bachmann & Bernstein, 2009), OpenOffice (Bachmann & Bernstein, 2009), FreeBSD (Spinellis, 2006), Windows Vista (Nagappan et al., 2008), Windows Server 2003 (Zimmermann &

Nagappan, 2008, 2009), Linux (Dempsey, Weiss, Jones, & Greenberg, 2002) and student projects (Dutoit & Bruegge, 1998; Edwards et al., 2006).

Commercial software organisations such as IBM and Microsoft seldom make their code history or project data publicly available. Thus, as is evident above, OSS repositories are often exploited to study process issues (Bachmann & Bernstein, 2009; Bird et al., 2006a). As noted previously, there are numerous challenges in using these repositories, related to the reliability and validity of the data available in these stores (Rodriguez et al., 2012). Research evidence has reported poor data quality in repositories of OSS projects (Aune et al., 2008; Bird et al., 2006a). In the Bird et al. (2006a) study of the Apache mailing list they found it difficult to uniquely identify developers' records due to the volume of email addresses and aliases these individuals used. Additionally, linking communication to source code entries presented many challenges for these researchers because of the different tools that were utilised during the project. For example, Subversion or Bugzilla may manage the code repository, while communication around software artefacts may be managed using a mailing list repository (Herzig & Zeller, 2009). Challenges associated with linking artefacts were also reported by Aune et al. (2008) in their study of Eclipse. Further issues may also be encountered when studying OSS repositories due to the way these projects are managed, because anyone is able to post messages and report bugs to such mailing lists, whether those individuals are contributing to the project or otherwise (Bettenburg et al., 2007; Bird et al., 2006a).

While careful use of data mining techniques may reduce the effects of these issues, threats to the validity and reliability of data analysed from these repositories are still likely to remain. In contrast, commercial data archives such as Jazz (studied in this work) have been reported to provide reliable process data (Bachmann & Bernstein, 2009; Nguyen, Wolf, et al., 2008). This is due to the use of well-defined and enforced software processes by these projects' host organisations. Additionally, since these datasets are not publicly accessible, only registered users are allowed to post messages and report bugs on such projects. In fact, during development at IBM Rational, project communication, the theme under investigation in this research, was enforced through the use of Jazz (Nguyen, Wolf, et al., 2008). Thus, the opportunity to study software human factors from this repository is invaluable, and provides an avenue for examining rich and reliable artefacts. Additionally, while previous studies have shown that communication among team members is affected in distributed software development

(see Herbsleb & Mockus (2003a) for example), a study examining this issue in the Jazz project has found no effect of distance on communication and project outcomes (Nguyen, Wolf, et al., 2008). This finding suggests that the Jazz environment effectively supported distributed software development, and evidence in this repository should indeed capture a comprehensive view of team processes.

However, in studying Jazz, one of the challenges relates to extracting and mining the data in this repository. Mining Jazz has been posited to be a highly complex activity, compared to mining OSS repositories (Herzig & Zeller, 2009; Nguyen, Schroter, & Damian, 2008). This complexity is linked to the tool's architectural and data storage requirements. Data mining principles have been generally recommended for addressing this challenge (Nguyen, Schroter, et al., 2008; Wolf, Schroter, Damian, & Nguyen, 2009), and so are utilised in this work. This issue is considered briefly in the following section (Section 3.4.2), a review of which informed the selection of specific techniques for accommodating the data extraction and pre-processing activities conducted during this study.

3.4.2 Data Extraction and Pre-processing - Data Mining

Data mining or knowledge mining is a discipline which combines statistics, artificial intelligence, pattern recognition and database management to facilitate knowledge extraction from large data sets (Han & Kamber, 2006; Tan et al., 2006). The motivation for the emergence and growth of this data mining resides in the fact that as computer hardware has become relatively inexpensive many industries have computerised their operations, which allows for the accumulation of massive data stores due to the many means available for rapid data collection (Tan et al., 2006). This abundance of data and the potential to gain knowledge by understanding patterns within (as illustrated in Figure 7 below), which may not be entirely evident through routine examination, have driven research into tools and techniques (considered under the data mining discipline) that facilitate knowledge extraction from large data sets (Larose, 2005).

Such tools and techniques have found utility in many organisations, for reasons including marketing research, population census trends, fraud detection and surveillance, science and engineering, games, traffic management, economics, health care and space research (Gorunescu, 2011; Han & Kamber, 2006; Tan et al., 2006). Under these areas data mining techniques have supported the prediction of future events and the description of data patterns (Han & Kamber, 2006). Prediction is normally

supported by classification and regression schemes, whereas clustering, association rule discovery and sequential pattern discovery provide descriptions of association and correlation (Gorunescu, 2011).

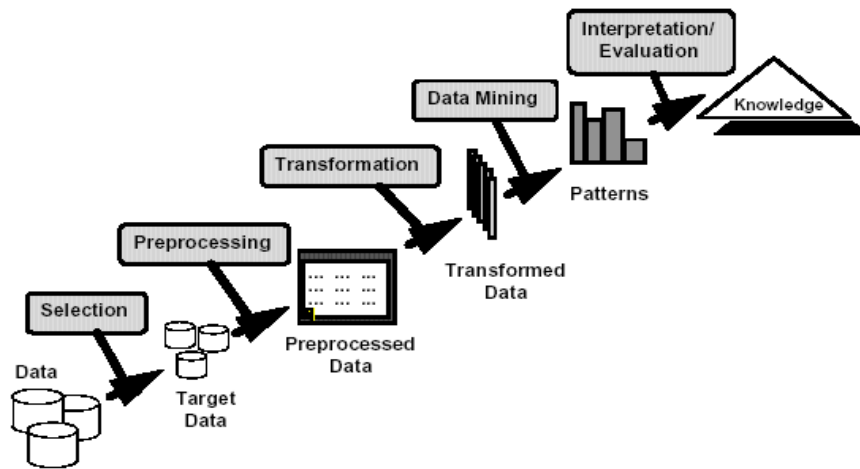


Figure 7. The data mining or knowledge mining process

For prediction and description measures to be accurate and provide high quality results, the data available for mining must be of a similarly high standard (Tan et al., 2006). In fact, applying data mining software such as Weka, KNIME, RapidMiner (these examples are freely available) to large data sets without employing data preparation measures to maintain data quality may lead to erroneous conclusions and potentially costly failures (Larose, 2005). Thus, in order to preserve or even enhance the standard and quality of data, which in turn may help in the discovery of more representative and valid results, many data pre-processing measures are employed. Data pre-processing, which has been described as the most challenging activity of the data mining exercise (Rodriguez et al., 2012), includes data cleaning, data integration and data transformation (Han & Kamber, 2006). Data cleaning involves detecting and removing noise from the data by modifying incorrect values and filling in missing values through estimation, or sometimes removing those instances or records altogether. Sometimes data cleaning may also involve identifying and removing outlier values (Tan et al., 2006). Data integration involves merging possibly many data sources into one large data source, and data transformation involves normalising and aggregating the data to improve the performance and accuracy of data mining algorithms.

In fact, data aggregation is often necessary in order to reduce the scale of the problem space so that mining algorithms perform efficiently. This activity potentially involves removing irrelevant data attributes, selecting a subset of the features, creating new

attributes that are better able to represent data objects, and changing data scales so that the smallest number of attributes are used during mining (Han & Kamber, 2006). Similarly, sampling may reduce the processing overhead required, while also allowing preliminary data exploration (Tan et al., 2006). When executing data mining algorithms, sampling could typically be of the forms: simple random sample (with or without replacement), cluster sample and stratified sample (Tan et al., 2006). For simple random sampling there is equal probability of selecting every record, while cluster sampling involves separating the repository into groups of records before applying the simple random sample method to select a number of clusters. Stratified sampling involves separating the data into partitions called strata, after which simple random sampling is employed on each stratum. Relevant artefacts may also be selected using purposive sampling strategies for more in-depth enquiries (refer to Section 3.4 for further discussion on purposive sampling).

These sampling techniques may be augmented by exploratory data analysis (EDA) (Larose, 2005). EDA denotes human involvement in the detection of patterns that are not necessarily captured by data analysis tools (Tukey, 1977). EDA utilises descriptive statistics, visualisation, clustering and anomaly detection (Tukey, 1977). Summary statistics may be used to investigate the general properties of the data, and particularly its distribution. Visualisation is the representation of data in visual form to facilitate analysis and identification of any relationships among attributes (Tan et al., 2006). Techniques for visualisation include histograms, box plots, scatter plots, contour plots, and matrix plots (Larose, 2005). Like summary statistics, computational methods such as cluster analysis and factor analysis can also be employed in data exploration to investigate correlation and multivariate measures. Multidimensional array representation (supporting On-Line Analytical Processing (OLAP)) may also be used for data analysis and presentation (Tan et al., 2006). All of these EDA techniques provide support for anomaly detection (Larose, 2005).

Although a complete investigation of data mining itself is beyond the scope of this work, specific aspects of data mining supported the activities involved in this research in terms of extracting, preparing and exploring the data under observation. In fact, it is largely recommended that data mining techniques be employed when extracting and preparing data from large repositories (Larose, 2005) – one of the activities undertaken in this work.

3.4.2.1 Data Extraction and Pre-processing Procedures

Data cleaning and transformation techniques (just mentioned) were utilized to enhance the representativeness of the data under consideration and to help with the assurance of data quality, while EDA techniques (including histograms and scatter plots) were employed to investigate data properties and for anomaly detection. As a result of these activities all records with inconsistent formats and data types were identified and removed, for example: integer columns with empty cells (resulting in the removal of 122 records from a total of 36,672). To verify that none of the 122 records were actually valid, these records were cross checked using the RTC (refer to Figure 8 for the typical view of a WI when accessed through the RTC), and this exercise validated that these records were not useful. Scripts were also written to remove all HTML tags and foreign characters from the textual data (including comments and work item descriptions).

The screenshot displays the Rational Team Concert (RTC) interface for a specific defect, titled "Defect 27647". The interface is divided into several sections:

- Summary:** A text box containing the defect title: "IFeedService isn't needed in IChangeEventRenderer methods". To the right of the text box are buttons for "Resolved", a dropdown arrow, and a checkmark.
- Details:** A section on the left containing various attributes:
 - Type: Defect (selected from a dropdown)
 - Severity: Normal (selected from a dropdown)
 - Found In: Unassigned (selected from a dropdown)
 - Created: 20/07/2007 10:18 AM
 - Created By: user2740
 - Team Area: Repository / Jazz Project
 - Filed Against: Repository/Gateway (selected from a dropdown)
 - Tags: (empty text box)
 - Owned By: user7372 (selected from a dropdown)
 - Priority: Medium (selected from a dropdown)
 - Planned For: 0.6 M4D1 (selected from a dropdown)
 - Estimate: 1 d, Correction: (empty text box)
 - Time Spent: (empty text box)
 - Due: 19/11/2007 (with a calendar icon)
 - Resolved: 23/10/2007 7:40 AM
 - Resolved By: user7372
- Description:** A text area containing the following text:

There is a visibility discrepancy between IChangeEventRenderer and IFeedService. IChangeEventRenderer.renderFeedEntry() takes an argument of type IFeedService, but IFeedService is in an internal repository package, yielding Discouraged Access warnings when one extends IChangeEventRenderer.
- Discussion:** A section titled "Discussion (5 comments; 6 new)" with an "Add Co" link. It contains a list of comments:
 - Comment 1: A code block starting with `/**` and containing several lines of JavaDoc-style comments and a `@param` tag that has been removed.
 - Comment 2: User user7372, 24/07/2007, 02:55. Text: "I'm assuming that it's OK to fix this for M3? Let me know if this needs to get fix earlier (M2)."
 - Comment 3: User user2740, 24/07/2007, 03:06. Text: "There is nothing urgent about it; I opened this work item as part of a sweep through all of SCM's 'Discouraged access' warnings, in an attempt to reduce those."
 - Comment 4: User user6339, 24/07/2007, 03:28. Text: "I changed the title, for us the only internal reference was caused by the param that wasn't needed by clients. So

Figure 8. Sample WI viewed via the RTC

A Java program was created to leverage the IBM Rational Jazz Client API to extract team information and development and communication artefacts from the Jazz repository. These included:

- Work Items (or Software Tasks) and history logs – in Jazz each software task is represented as a WI (refer to Section 3.4.1 for details), and a change history log is maintained for each WI.
- Project Workspaces or Project Areas – each Jazz team is assigned a workspace (also called a project or team area). The workspace contains all the artifacts belonging to the specific team.
- Contributors and Teams – a contributor is a practitioner contributing to one or more software features, multiple contributors form teams. For the actual role information extracted from the repository, the Team leads (component lead) are responsible for planning and architectural integration of components. Admins (including roles related to integration, administration and configuration) are responsible for configuration and integration of artifacts. Project managers (PMC) are responsible for project governance. Those occupying the Programmer (contributor) role contribute code to features. Finally, those that occupied more than one of these roles were labeled Multiple.
- Comments or Messages – communication around WIs is facilitated by Jazz's comment functionality. Messages ranged from as short as one word (such as 'thanks'), to up to 1055 words representing multiple pages of communication. These are arranged by date sequentially for each WI, similar to messages on a bulletin board (see Figure 8 for illustration).

A total of 36,672 resolved WIs (36,550 pre-processed) and history logs created between June, 2005 and June, 2008 were extracted from the Jazz repository. These work items belonged to 94 project workspaces that each comprised more than 25 WIs. This volume of data was considered sufficient to support the study's investigations. The project workspaces contained the work of 474 active contributors belonging to the five different roles noted above. For the 94 project areas, comments (or messages) – the primary data source for this research – were also extracted, totalling 116,020. The data extracted from Jazz were imported into a Microsoft SQL relational database management system to facilitate efficient data management (see the database model of the pre-processed and partially normalized Jazz data in Figure 9). Although there was some redundancy in the Work Item table that was created (for some fields including: Type, Severity, Priority, and so on - see Figure 8 for the full list of WI fields), this was not an issue of concern in this research as there was no need to conduct additions, deletions or modifications (or

any performance-related transactions) to any of the records stored in this table. Thus, further division of the Work Item table into smaller tables (to reduce redundancy) was not necessary.

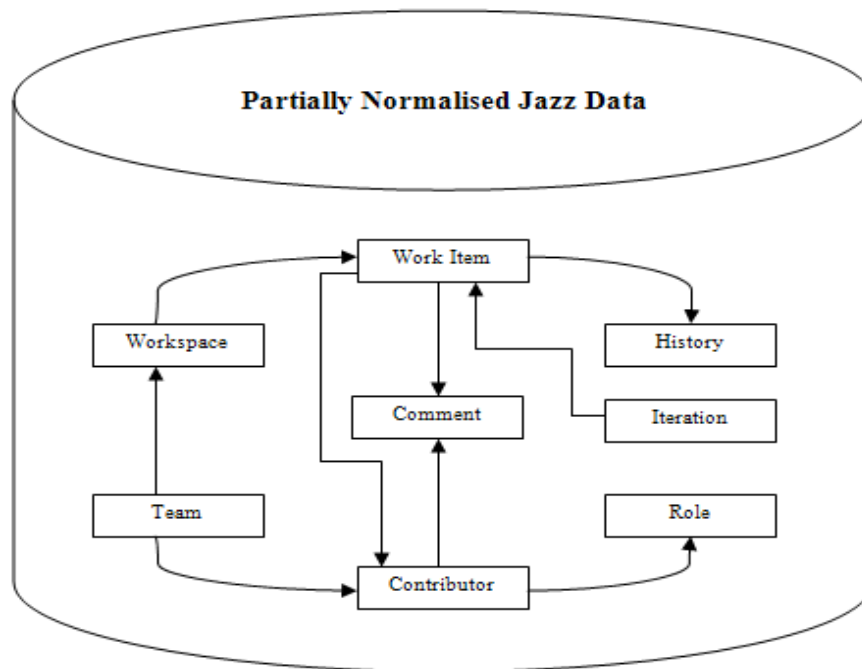


Figure 9. Database model of the pre-processed and partially normalized Jazz data

In line with the multi-case study design outlined above, purposive sampling was used in the case selection process (Yin, 2003). The goal was to select a range of cases that represented the scope and breadth of the various teams in the repository, for example: some team areas are labelled as documentation, user experience, development or coding, and project management-based activities. Thus, initially all the tasks (WIs) undertaken by ten of the 94 project teams (shown in Table 2) were selected for examination. The team areas selected represent both information-rich and information-rare cases in terms of numbers of messages (refer to Table 2). These cases also represent development activities (planned in multiple iterations) that were of varied durations, from short (59 days or two iterations) to long (1014 days or 17 iterations), with varying levels of communication density. Selected teams' artefacts amounted to 1201 software development tasks, carried out by 394 contributors (and comprising 146 distinct members from the 474 total contributors), with 5563 messages exchanged around the 1201 tasks.

Social network analysis measures and graphs were also used to inform the data sampling, and particularly for studying the patterns of interaction in order to inform data saturation (refer to Section 3.4.3 for details). For the artefacts from the ten teams, during

social network analysis it became clear that the cases selected were representative of those in the repository in relation to team members' communication and engagement in task (feature) changes. Project communication and engagement for all ten teams was heavily skewed around only few members, and data saturation was achieved after analysing the team artefacts in the third case (Glaser & Strauss, 1967; Licorish & MacDonell, 2013d) (see Section 3.4.3 and Appendix II for further details). This initial pattern of concentration around a few individuals in each team workspace provided preliminary confirmation for the wheel structure pattern regularly noted in Chapter 2, and set the tone for the particular detailed analysis of core developers.

Beyond the data saturation observed, the sample of (ten) teams' artefacts selected is considered to be adequate for the type of inquiries conducted during this work (Creswell, 1998), particularly as all the study cases were selected from a single data source (Kuzel, 1992), and multiple analysis approaches (both top-down and contextual) were applied to a large number of messages (5563 altogether). Additionally, Romney, Weller, & Batchelder (1986) showed that samples representing the work of just four individuals (the sample in this study comprised 146 distinct practitioners) could render highly accurate information if those individuals were very competent in the domain under investigation, as it was posited for Jazz developers given the high level of their tools' usage and the positive reviews these tools received (noted in Section 3.4.1).

Data collected are analysed using social network analysis techniques, linguistic procedures, and content analysis processes, in a multi-phase approach (refer to Section 3.4). (Note that SNA was also used to inform the case selection process and to confirm that the cases selected were adequate.) These aspects of the research design are considered in the following three sections (Sections 3.4.3, Section 3.4.4 and Section 3.4.5 respectively), starting with a review of the SNA techniques in Section 3.4.3.

Table 2. Summary statistics for the selected Jazz teams

Team ID	Task (WI) Count	Software Tasks (Project/Team Area)	Total Contributors – Roles	Total Messages	Period (days) – Iterations
P1	54	User Experience – tasks related to UI development	33 – 18 programmers, 11 team leads, 2 project managers, 1 admin, 1 multiple roles	460	304 - 04
P2	112	User Experience – tasks related to UI development	47 – 24 programmers, 14 team leads, 2 project managers, 1 admin, 6 multiple roles	975	630 - 11
P3	30	Documentation – tasks related to Web portal documentation	29 – 12 programmers, 10 team leads, 4 project managers, 1 admin, 2 multiple roles	158	59 - 02
P4	214	Code (Functionality) – tasks related to development of application middleware	39 – 20 programmers, 11 team leads, 2 project managers, 2 admins, 4 multiple roles	883	539 - 06
P5	122	Code (Functionality) – tasks related to development of application middleware	48 – 23 programmers, 14 team leads, 4 project managers, 1 admin, 6 multiple roles	539	1014 - 17
P6	111	Code (Functionality) – tasks related to development of application middleware	25 – 11 programmers, 9 team leads, 2 project managers, 3 multiple roles	553	224 - 13
P7	91	Code (Functionality) – tasks related to development of application middleware	16 – 6 programmers, 7 team leads, 1 project manager, 1 admin, 1 multiple roles	489	360 - 11
P8	210	Project Management – tasks under the project managers' control	90 – 29 programmers, 24 team leads, 6 project managers, 2 admins, 29 multiple roles	612	660 - 16
P9	50	Code (Functionality) – tasks related to development of application middleware	19 – 10 programmers, 3 team leads, 4 project managers, 2 multiple roles	254	390 - 10
P10	207	Code (Functionality) – tasks related to development of application middleware	48 – 22 programmers, 12 team leads, 2 project managers, 1 admin, 11 multiple roles	640	520 - 11
Σ	1201		394 contributors , comprising 175 programmers, 115 team leads, 29 project managers, 10 admins, 65 multiple roles	5563	

3.4.3 Data Analysis (Technique 1) - Social Network Analysis

Social Network Analysis (SNA) may be used to quantify aspects of network structures in order to support pattern identification in social networks (De Laat et al., 2007). This technique employs mathematical analysis and pictorial representations of the patterns of interaction and relationships among individuals – and potentially other components – during group processes (Willging, 2005). Concepts such as cohesion, equivalence, power and brokerage are used to explain the characteristics of network actors (Scott, 2000). Of these concepts, the most important mathematical measurement for SNA is cohesion, measured by density and centrality. Density provides an overall measurement of the connectedness of the network (Scott, 2000), whereas centrality (also called degree or degree centrality) denotes the level of individual interaction (Wasserman & Faust, 1997). Visualisation of interaction networks, also called sociograms, is often used for uncovering interaction patterns and the flow of information that may not be so evident

from numerical values (Wasserman & Faust, 1997). In these visualisations, individuals are represented by nodes, and their associations are illustrated through lines that connect these nodes. An examination of a sociogram will unveil who is communicating (or not), who is most central to the team, which members are acting as hubs or brokers, and so on.

Figure 10 shows a simple sociogram with four different network segments (a, b, c, and d) to illustrate how visualisations support the detection of interaction patterns among individuals. In Figure 10(a) the solid blue member has a degree centrality of 6 because (s)he is directly connected to 6 other individuals. When compared to the other individuals in this network segment (whose degree centrality ranges between 2 and 3), this is the most central member in this group of communicators. In Figure 10(b) the solid green node (member) forms the link (hub or bridge) between two network segments (Figure 10(a) and Figure 10(c)), while in Figure 10(c) the solid red node, although playing the role of bridge for segments Figure 10(c) and Figure 10(d), has a degree centrality of 1 in Figure 10(c), and is the weakest communicator in both Figure 10(c) and Figure 10(d). The most dense network segment is seen in Figure 10(d) where all members have a degree centrality of between 5 and 6 (with 6 being the highest possible degree centrality). In this network segment (Figure 10(d)), the solid black member has a density of 1 (the maximum value for the density measurement as further described in Section 3.4.3.1).

SNA has been shown to have value in many domains, including security (Dekker, 2002), political science (Mendieta, Schmidt, & Ruiz, 1997), education and communication (Aviv, Erlich, Ravid, & Geva, 2003), as well as in software engineering (Cataldo et al., 2006; Wolf, Schroter, Damian, Panjer, et al., 2009). While it is generally recommended that caution be shown when explaining the consequences of social network patterns due to the many reasons people may interact (Aral & Walker, 2011) – whether the reason be due to peer authority, social pollution or otherwise – collaboration in professional software engineering settings is generally linked to work task execution, and active collaboration has been shown to have a positive impact on team productivity (Damian & Zowghi, 2003; Ehrlich & Chang, 2006). Thus, SNA techniques provided utility for this work, and are used to identify unique interaction patterns among software tasks and key software practitioners. Apart from confirming previous evidence, these techniques also informed the other stages of the data analysis

(refer to Section 3.4.4 and Section 3.4.5), and the case selection process (noted in Section 3.4.2.1).

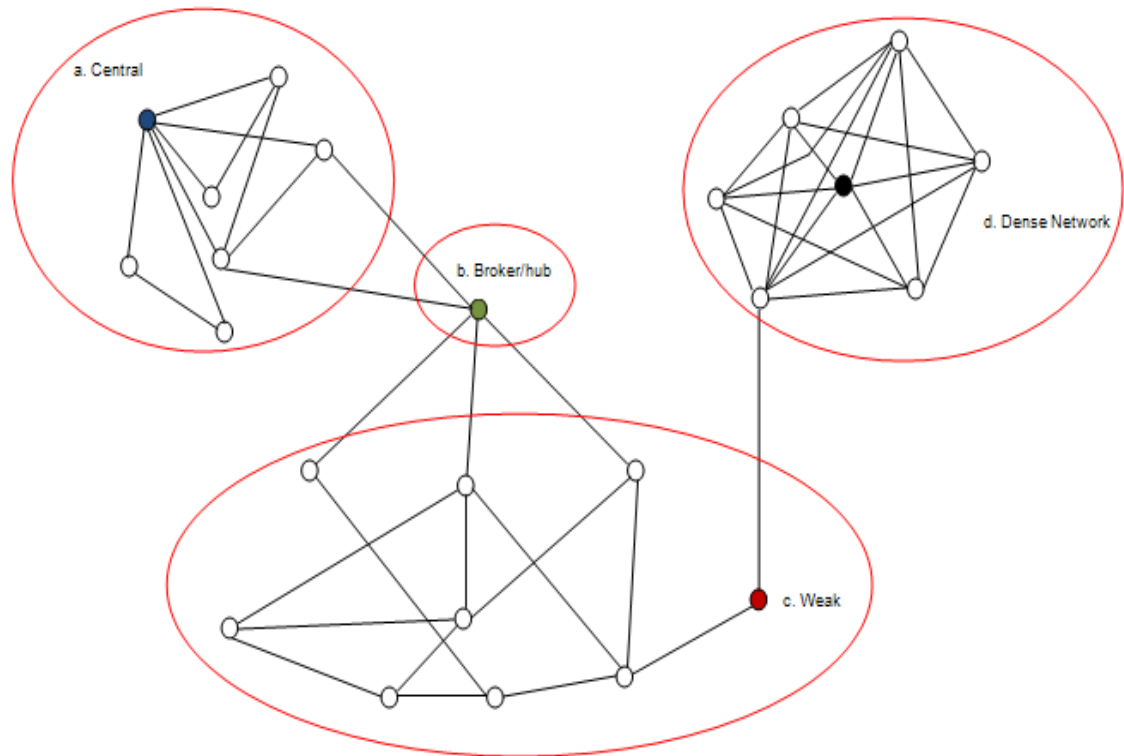


Figure 10. Sociogram highlighting interaction patterns of team members

In line with this study's units of analysis outlined above, interaction patterns around individuals and teams are of primary interest during the SNA observations. These units are central to the way the patterns identified are interpreted. Firstly, a task-based sociogram was created from all the tasks and comments extracted from the Jazz repository during the data extraction and pre-processing phase (refer to Section 3.4.2 above for further explanation of the data extraction and pre-processing activities that were conducted). For this social network a software task represented a unit of work, and the messages communicated by the practitioners solving the software task represented the collaboration and interaction that was aimed at completing the specific task. During the construction of the Jazz project sociogram, tasks (WIs) and comments formed the network nodes, and edges connected these nodes – such that an individual commenting on a software task represented a simple graph with two nodes and one edge; see Wolf, Schroter, Damian, Panjer et al. (2009) for further reading on constructing task-based social networks. By qualitatively examining these Jazz network visualisations, it was indeed confirmed that the Jazz repository was partitioned based on teams areas (see Figure 11 for a section of the overall Jazz sociogram with visualisations of teams' and practitioners' interaction patterns), and various members acted as hubs across the

different teams which made the overall Jazz network highly connected. Quantitative measurements confirmed the variations in connection density around various network segments noted through qualitative observations (and these are presented in Chapter 4). This evidence informed the data pre-processing and data normalisation process, and supported preliminary observations made during the execution of standard queries and EDA (noted in Section 3.4.2).

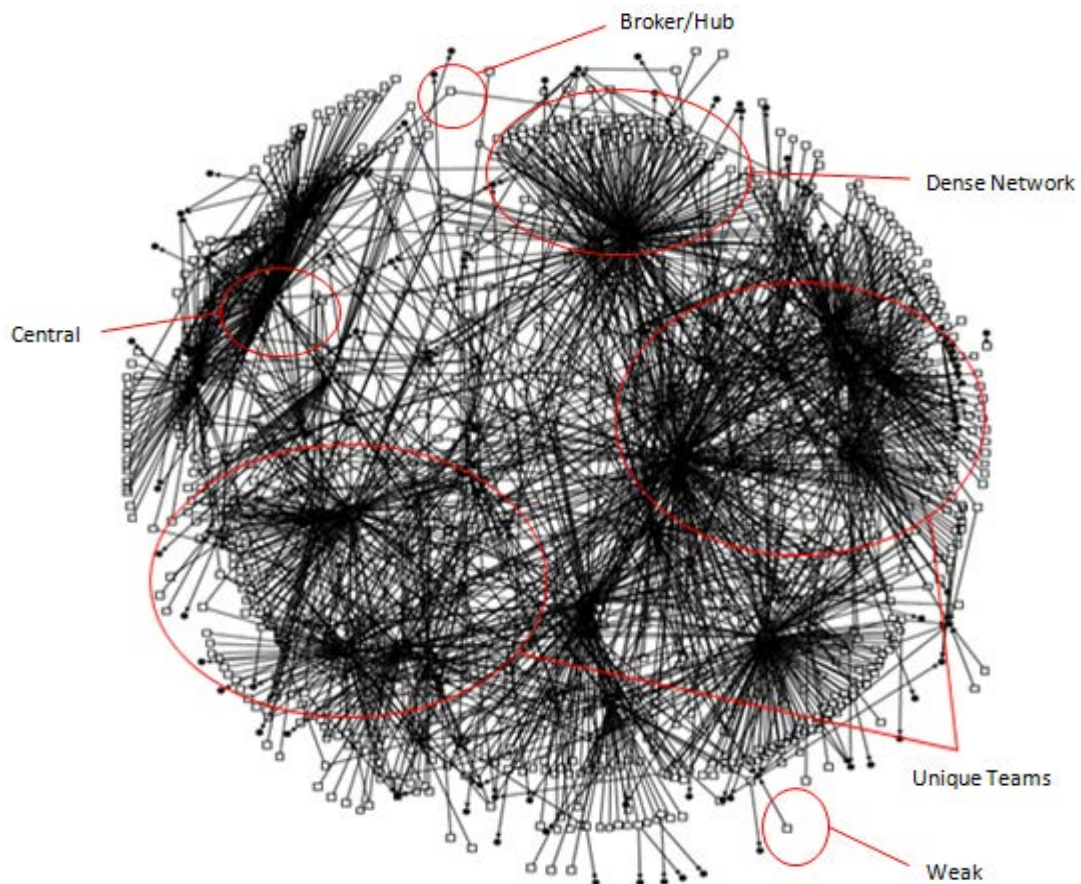


Figure 11. Section of Jazz communication network

Given these preliminary observations coupled with the ability to query the Jazz data by team area (as per the database model shown in Figure 9), sociograms were created that represented all the tasks addressed by specific teams and the communication around these tasks. However, finer grained qualitative examination of these graphs proved challenging due to the volume of network connections in each team area. This issue was encountered previously by those examining graph readability (Ghoniem, Fekete, & Castagliola, 2005; Henry & Fekete, 2007). In order to make the network visualisations more meaningful, the teams' sociograms were reconstructed using directed social networks, and network edges belonging to distinct contributors on individual software tasks were merged and colour coded; edge colour moved from red to brown (between one to five messages), brown to green (between six to ten messages) and then to a more

pronounced green (eleven or more messages). The network vertices also represented either a class image denoting a task or a contributor's unique identification number (see Figure 12 for a sample Jazz sociogram that was created using this procedure). Constructing the teams' aggregated networks in this way does not represent a limitation to the study design as the goal of this aspect of the work was to identify unique interaction patterns among software tasks and key software practitioners, and information flow and interaction in the Jazz environment (or indeed any real world project) is directed (Datta et al., 2010).

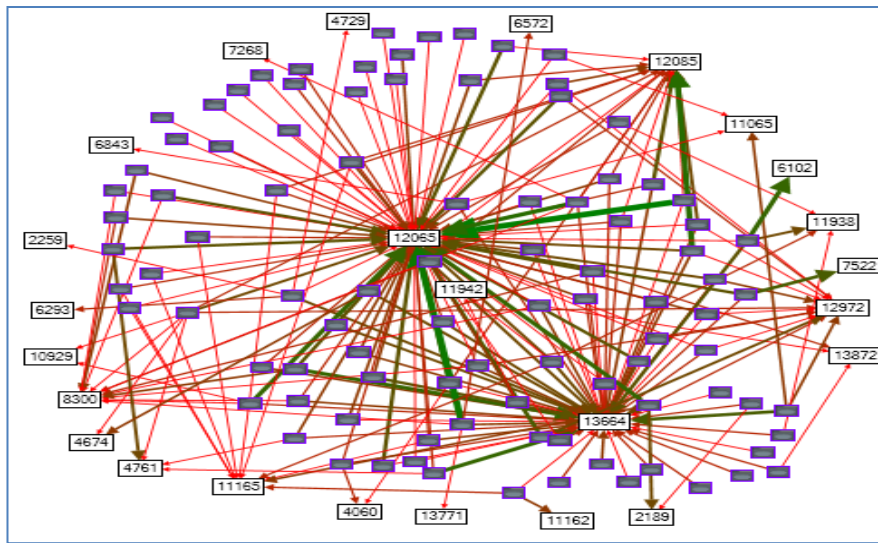


Figure 12. Directed network graph for a sample Jazz team showing highly dense network segments for practitioners “12065” and “13664”

3.4.3.1 SNA and Other Quantitative Measures and Procedures

Apart from the visualisation of individuals' and teams' interactions using sociograms, other complementary SNA measures (introduced above) were also utilised to study Jazz teams' collaboration patterns. These measures and procedures are as follows:

- **Density** varies between 0 and 1, so that a task or individual that attracted interaction from all the members in a team would have a density of 1, while those with no interaction would have a density of 0. The network density measure is used to study the team networks' connectedness and practitioners' level of interactions during Jazz development (e.g., a practitioner that communicated on 20 out of their team's 50 tasks would have a density of 0.4). Individuals involved in highly dense communication network segments have been shown to dominate coordination and collective action (Reagans & Zuckerman, 2001), and are seen as most important to their teams (Zhong, Huang, Davison, Yang, & Chen, 2012).

This measure is also used to select core developers, as was done in previous work (Crowston et al., 2006). To this end a classification threshold is employed following a similar approach as used by (Crowston et al., 2006), so all contributors whose density measure was ≥ 0.33 (i.e., they communicated on a third or more of their team's tasks) are selected and grouped as 'top contributors' or 'core developers'.

- **Centrality** measures for **In-degree** and **Closeness** are used to study how connected teams and their contributors' were during their project and levels of interaction. These measures are also use to assess how accessible individuals were in the teams' networks, as has been considered by others (Bird et al., 2006a; Datta et al., 2010). In-degree denotes the number of connections that point towards a vertex and is used to provide measures for the number of unique messages (edges) generated by individuals during Jazz development. Practitioners' numbers of unique messages (edges) in their network graphs are also aggregated to provide a measure for the number of distinctive contributors to communication during the teams' developments. Closeness measures the shortest distance between nodes. While in-degree accounts for the immediate links around a vertex, closeness measures both direct and indirect connections (Datta et al., 2010; Hanneman & Riddle, 2005). So, the lower the closeness measure for a given node the more reachable that node is to the other members in the network (Hanneman & Riddle, 2005). Such nodes are likely to play an important role in maintaining network connectivity, and are generally regarded as powerful (Hanneman & Riddle, 2005). This measure is used to evaluate the strength and accessibility of the teams' networks and of individual team members.
- Various approaches have been used over many years to measure individual-level **performance in software tasks**. Productivity-related measures such as lines of code per unit of effort (Curtis, 1981), time taken to complete development tasks (Espinosa et al., 2007) and the number of task changes completed (Cataldo & Herbsleb, 2008) are among those used previously to measure performance. Among others, Cataldo and Herbsleb (2008) argued that measures based on lines of code may not be reliable in instances where there is variability in developers' coding styles (i.e., some developers are more verbose than others). The time taken to complete development tasks may vary for developers when there are many feature inter-dependencies (e.g., a developer may start working on a feature that

needs to use classes that are under development by another developer, and thus may be delayed). Therefore, this work use the relative number of task (WI) changes as indicative of the performance of developers in software tasks (Cataldo & Herbsleb, 2008; Shihab et al., 2010). A developer was considered to change a task if they created, modified, or resolved that task, as has been utilised in prior studies (Cataldo & Herbsleb, 2008; Ehrlich & Cataldo, 2012).

- From the cases selected it is noted that software tasks were planned in multiple iterations for each team area (P1 – P10 in Table 2), and for further discussion around the way the Jazz project is organized see Ehrlich & Cataldo (2012). However, Table 2 also shows that the actual number of iterations varied across the team areas (e.g., P3 tasks were completed in two iterations, whereas P5 tasks were executed in 17 iterations). Therefore, to normalise the teams' task data in order to examine any changes in practitioners' **interactions over project duration**, each team's tasks and artefacts are divided into four equal quarters (start, early-mid, late-mid, and end) (Licorish & MacDonell, 2012, 2013d).
- The various team areas (and software tasks) are used to uniquely identify the **nature of the software development activities**; e.g., those working on P1 and P2 in Table 2 were undertaking tasks that are labeled as User Experience related functionalities (Licorish & MacDonell, 2013d).
- **Formal team role** information (e.g., Programmer, Team lead, Project manager and Admin) and data regarding practitioners' responsibilities extracted from the Jazz repository are used in comparing practitioners' formal roles with their involvement in their teams' interactions, their enacted roles and task engagement (Licorish & MacDonell, 2013c).

The NodeXL tool is used to support the SNA-related aspects of this study, enabling the production of network visualisations and the calculation of the metrics that are introduced above. NodeXL is an open source software tool used to model data drawn from social media sources, such as email, discussion forums, wikis and blogs, to support the understanding of interactions and relationships created through the use of these media (Hansen, Shneiderman, & Smith, 2011; Smith et al., 2009). Other such tools include Unicet and NetDraw (see <http://www.insna.org> for other SNA tools). NodeXL is widely used for constructing sociograms and studying communication and interaction

(Datta et al., 2011; Sharma & Kaulgud, 2011), in line with the research objectives of this study.

While of interest and utility in their own right, the results obtained from the SNA are also triangulated through deeper linguistic analysis. The mechanisms and frameworks that are used in this regard are reviewed next. The next section (Section 3.4.4) is dedicated to introducing linguistic analysis techniques and outlining how a specific approach is selected for use during this research.

3.4.4 Data Analysis (Technique 2) - Linguistic Analysis

It has been established in the previous chapter that linguistic analysis of textual communication can reveal much about those communicating. This provides a rationale for the analysis of Jazz project developers' messages in this research. There are many candidate approaches for analysing attitudes and behaviours from textual data. In reviewing the literature in the psycholinguistic space, it is observed that the Linguistic Inquiry and Word Count (LIWC) software and the Medical Research Council (MRC) Psycholinguistic Database are most frequently adopted for this form of analysis (Coltheart, 1981; Gill & Oberlander, 2003; Mairesse et al., 2007). Both utilise established dictionaries in the text analysis process and so may be regarded as top-down approaches (Nowson & Oberlander, 2006). In addition to the LIWC tool and MRC database, there are more qualitative techniques that may be deemed data driven or bottom-up approaches. These data driven approaches rely on the data itself to provide the dimensions for classification, as against grouping data based on already available feature sets or dictionaries, as found in top-down approaches. These data driven approaches are said to capture the specific context in which words are used, often via n-grams (Damerau, 1993), and statistical analysis is used to test for significance.

Given that this study is aimed at examining communication artefacts to investigate software practitioners' attitudes as evident in 5563 messages, utilising a data driven approach is not practical due to its resource-intensive nature. More importantly, this study is not intended to create psycholinguistic theories, but rather to explore the attitudes and behavioural patterns evident during successful agile globally distributed software development activities. Further, previous studies verifying individuals' attitudes and behavioural issues using textual communication have successfully employed dictionary-based approaches (Coltheart, 1981; Rigby & Hassan, 2007). Thus,

the most frequently used top-down approaches are reviewed here to justify the selection of the specific technique that is employed during this research.

As noted above, one of the most frequently employed linguistic analysis approaches, the LIWC, is a software tool created after four decades of research using data collected across the USA, Canada and New Zealand (Pennebaker et al., 2007; Pennebaker & King, 1999). Data collected in creating the LIWC tool spanned many areas of life, including emotional writing, control writing, research articles, blogs, novels and normal conversations (and data collection is an on-going exercise). This tool captures over 86% of the words used during conversations (comprising around 4500 words) and is available in many languages. In the tool, words are grouped based on specific types, such as negative emotion, social words, positive emotion, quantifiers, and so on (refer to Table 3 below (Pennebaker et al., 2007) for a sample of the tool's linguistic categories). Written text is submitted as input to the tool in a file which is then processed and summarised based on the LIWC dictionary. Each word in the file is searched for in the LIWC dictionary, and specific scales are incremented in accordance with the word category, after which a file is returned to the user containing the summary output. The tool's output data include the percentage of words captured by the dictionary, standard linguistic dimensions (e.g., pronouns and negation), psychological categories and function words (e.g., negative and social) and personal dimensions (e.g., work and leisure).

The MRC database, on the other hand, contains psycholinguistic information for around 9240 imagery rating words (Coltheart, 1981; Wilson, 1987). This tool works in a similar way to the LIWC, scoring text for the numbers of letters, phonemes and syllables, words written in specific categories, sample counts, verbal frequency, concreteness, imaginative words, and others, as derived from the 1963 Oxford dictionary and Edinburgh Associative Thesaurus. As with the LIWC, this tool returns summary statistics including the mean and standard deviation of the variables just described for the input text. The MRC database was designed to support psycholinguistic, language processing and cognitive simulation research, with its dictionary comprising a number of small language databases (see the Sample of the dictionary file in Table 4, (Coltheart, 1981; Wilson, 1988)).

Table 3. Sample LIWC output variable information

Category	Abbreviation	Examples	Words in Category
Linguistic Processes			
1st pers singular	i	I, me, mine	12
1st pers plural	we	We, us, our	12
2nd person	you	You, your, thou	20
Prepositions	prep	To, with, above	60
Conjunctions	conj	And, but, whereas	28
Negations	negate	No, not, never	57
Quantifiers	quant	Few, many, much	89
Personal Concerns			
Work	work	Job, majors, xerox	327
Achievement	achieve	Earn, hero, win	186
Leisure	leisure	Cook, chat, movie	229
Home	home	Apartment, kitchen, family	93
Psychological Processes			
Social processes	social	Mate, talk, they, child	455
Positive emotion	posemo	Love, nice, sweet	406
Negative emotion	negemo	Hurt, ugly, nasty	499
Anxiety	anx	Worried, fearful, nervous	91
Anger	anger	Hate, kill, annoyed	184

Table 4. Sample MRC dictionary file

Column	Name	Property	Occur
41-43	AOA	Age of Acquisition	3503
44	TQ2	Type	44976
45	WTYPE	Part of Speech	150769
46	PDWTYPE PD	Part of Speech	38390
47	ALPHSYL	Alphasyllable	15938
48	STATUS	Status	89550
49	VAR	Variant Phoneme	1445
50	CAP	Written Capitalised	4585
51	IRREG	Irregular Plural	23111
	WORD	the actual word	150837
	PHON	Phonetic Transcription	38420
	DPHON	Edited Phonetic Transcription	136982
	STRESS	Stress Pattern	38390

According to linguistic theories, it is possible to discern attitudes within individuals' textual communications (Giles & Wiemann, 1993; Hart, 1984; Oxman et al., 1988; Pennebaker & King, 1999; Pennebaker et al., 1997; Pennebaker et al., 2003; Schnurr et al., 1992; Spence et al., 1978; Taylor et al., 1994). As noted previously, extensive prior research has revealed that an individual's linguistic style is quite stable over time, and that text analysis programs are able to accurately link language characteristics to individual behaviours (Mairesse & Walker, 2006; Mairesse et al., 2007; Pennebaker &

King, 1999; Pennebaker & Lay, 2002). Language use has also been studied as a function of age (Pennebaker & Stone, 2003), gender (Mulac et al., 2001) and emotional upheaval (Stone & Pennebaker, 2002). In sum, these studies provide compelling evidence that language use is contextual, such that the way in which individuals communicate is influenced by their context and local settings. While previous studies have also found correlation between individuals' language use and their attitudes through the use of the LIWC and MRC tools (Mairesse et al., 2007; Yee, Harris, Jabon, & Bailenson, 2010), research has found the LIWC scales to be more accurate than those of the MRC in support of assessing behaviours from communication (Mairesse & Walker, 2006). In addition, the LIWC has also found wider support in terms of being linked to attitudes in the psycholinguistic literature, when compared to the MRC database (Li & Chignell, 2010; Mairesse et al., 2007; Yee et al., 2010). Consequently, this study has employed this tool to examine software practitioners' attitudes and behaviours based on the language expressed in their messages.

Immediate benefits provided to this study by using the LIWC tool are related to validity and reliability. As noted previously, studies in the psycholinguistic area have repeatedly tested this instrument for analysing behaviours from text (Mairesse et al., 2007; Pennebaker & King, 1999). Additionally, findings from enquiries undertaken in this study assess the suitability of the LIWC tool for analysing software practitioners' textual communication data. The specific measures that are selected for studying software practitioners' behaviours and attitudes are set out in Section 3.4.4.1, and these align with the study focus and questions described in Chapter 2. The procedures utilised during this study are subsequently introduced in Section 3.4.4.2.

3.4.4.1 Behaviour and Attitude Analysis Measures

In terms of behaviours and attitudes, the LIWC tool's various dimensions are said to capture the psychology of individuals by assessing the types of words they use (Pennebaker et al., 2007; Pennebaker & King, 1999). In fact, although it may be argued that counts of words may not capture the *specific* contextual meaning of word usage (Krauss & Fussell, 1996; Zeldow & McAdams, 1993), when taken together the usage of such words can indicate an individual's temperament, their language composition preferences, their psychological traits (Pennebaker & King, 1999) and their moods (Denning, 2012). For example, consider the following two sample comments:

1. “I dislike the way the customer service team works, especially the delay they cause me. This delay will no doubt affect my overall performance-appraisal when I am assessed towards the end of the year.”
2. “We are aiming to have all the patches ready by the end of this release; this will provide us some space for the next one. Also, we are extremely confident that similar bug-issues will not appear in the future.”

In the first comment the author is expressing dissatisfaction at the treatment received from another department, and is worried about the potential negative consequences this treatment will cause. Here the words “I”, “my” and “me” are indicators of self-focus, words such as “dislike” are associated with negative feelings, and “end” denotes some form of temporal reference. Words such as “performance-appraisal” are *not* captured by the LIWC dictionary, and as such the summary output for the text above would be the same whether the author was referring to achievement at sports, or work on a software feature. Although such omissions may be interpreted as presenting a limitation, this is not the case here given that the study context is known to be software development; and what *is* of interest, and *is* being captured by the tool, is evidence of attitude, demeanour and behaviour.

In the second comment the author is expressing optimism that the team will succeed, and in the process finish ahead of time and with acceptable quality standards. In this quotation, the words “we” and “us” are indicators of team or collective focus, “all”, “extremely” and “confident” are associated with certainty, while words such as “some” and “appear” are indicators of tentative processes. As in the use of the words “performance-appraisal” in the first comment, words such as “bug-issues” and “patches” are not included in the LIWC dictionary and would not affect the context of its use – whether it was to indicate a fault in software code or a problem with one’s immunity to a disease. Thus, in terms of the assessment of practitioners’ behaviours and competencies, the LIWC tool’s output is not adversely affected by the specialized nature of the vocabulary associated with a specific discipline, or the specific meaning with which these words are used.

Rigby & Hassan (2007) have demonstrated the utility of the LIWC tool during their study of developers’ messages from the Apache OSS mailing list. Among their findings, these authors’ revealed major changes in the top two developers’ attitudes once they decided to leave the Apache project. Additionally, the wider team of Apache developers

also expressed different attitudes over project releases (Rigby & Hassan, 2007), at one time expressing more optimism and being more cynical on another occasion.

During the early stages of this research the LIWC tool was also used in a preliminary study of three different Jazz teams to reveal cues for the ways these different teams might work (Licorish & MacDonell, 2012). In this study variance in behaviours was found among those undertaking different forms of software tasks, which it was believed may be linked to differences in the project portfolio. However, this initial study was largely exploratory and only a small sample of artefacts were examined, which limited this study's inferences (Licorish & MacDonell, 2012).

Whereas Rigby and Hassan (2007) looked at the behaviour expressed in the language of the top four contributors of the Apache project and also the team's behaviour after two releases, core developers' behaviours and attitudes are studied in this research and are compared to those of their lesser active counterparts. This work also examines whether core developers' attitudes change over the duration of their project, and how these members' attitudes are linked to their task performance. Behaviours and attitudes are examined along multiple linguistic dimensions (provided in Table 5). Furthermore, artefacts that are studied in this research are taken from a representative software repository, comprising a total of 146 unique contributors (from 474 total Jazz practitioners), 5563 messages and over 1200 software tasks (WIs) and associated history logs (refer to Table 2). Table 5 provides a summary of the LIWC linguistic categories that are considered during this research, along with brief theoretical justifications (with some also included in Chapter 2) for their inclusion.

Table 5. LIWC linguistic measures

Linguistic Category	Abbreviation (Abbrev.)	Examples	Reason for Inclusion
Pronouns	I	I, me, mine, my, I'll, I've, myself, I'm	Individuals favouring more collective group processes may demonstrate this trait through their language use (Pennebaker & Lay, 2002). Previous research has found elevated use of first person plural pronouns (we) during shared situations and among individuals that share close relationships, whereas, relatively high use of self-references (I) has been linked to individualistic attitudes (Pennebaker et al., 2003; Stone & Pennebaker, 2002). First person singular and plural pronoun linguistic dimensions are considered here to represent self-focused attitudes or shared group processes among members. Use of the second person pronoun (you) may signal the degree to which members rely on (or delegate to) other team members or their general awareness (Pennebaker et al., 2003) of others and their activities.
	we	we, us, our, we've, lets, we'd, we're, we'll	
	you	you, your, you'll, you've, y'all, you'd, yours, you're	
Cognitive language	insight	think, believe, consider, determined, admitted, idea	Software teams were previously found to be most successful when many group members were highly cognitive and were natural solution providers (Andre et al., 2011). These traits also previously correlated with effective task analysis and brainstorming capabilities. These linguistic dimensions are included so that this work can analyse communication artefacts to assess the cognitive aspects of team members.
	discrep	should, prefer, needed, problem, regardless,	
	tentat	maybe, perhaps, apparently, chance, appears, hopeful	
	certain	definitely, always, extremely, absolute, certain	
Work and Achievement related language	work	feedback, goal, boss, overtime, program, delegate, feedback, duty, meeting	Individuals most concerned with task completion and achievement are said to reflect these traits during their communication. These individuals are most concerned with task success, contributing and initiating ideas and knowledge towards task completion (Benne & Sheats, 1948). Work- and achievement-related communications are analysed to assess those most concerned with task completion.
	achieve	accomplish, attain, closure, resolve, obtain, finalize, fulfil, overcome, solve	
Leisure, social and positive language	leisure	club, movie, cinema, entertain, gym, golf, party, jog, film	Assessment of the use of leisure terms, the opposite to work, is used to measure the relative frequency of off-task interactions within teams. Individuals that are personal and social in nature are said to communicate positive emotion and social words and this trait is said to contribute towards an optimistic group climate, promoting encouraging, harmonizing and compromising traits (Benne & Sheats, 1948; Zhu, 1996). These linguistic dimensions are used to study those that express social team behaviours (van den Hoof et al., 2004).
	social	give, buddy, love, explain, friend, hey, inform, meet, pal	
	posemo	beautiful, relax, perfect, glad, eager, fantastic, luck, impress, proud	
Negative language	negemo	afraid, bitch, hate, suck, dislike, shock, sorry, stupid, terrified	Negative emotion may affect team cohesiveness and group climate. Those expressing significant negative emotion are also said to have a tendency to be unfulfilled or dissatisfied and to show excessive anger (Denning, 2012; Goldberg, 1981). This linguistic dimension is used to study those members that contribute negatively to team behavioural climate (Chang et al., 2013; Denning, 2012).

3.4.4.2 Linguistic Analysis Procedures

In order to analyse practitioners' behaviours and attitudes, all of the 5563 comments in Table 2 were first exported from Microsoft SQL into a text (.txt) file. A Java program was then created to parse the exported text file, creating new text files for each of the comments (so there were now 5563 new files). The LIWC program was then executed through a batch call, using all 5563 new files as input. These files were then analysed in terms of the linguistic dimensions in Table 5, and returned in one LIWC result archive. The LIWC result archive was then imported into the Microsoft SQL relational database,

where each comment was linked to its corresponding LIWC analysis record, extending the database design (refer to Figure 9) as shown in Figure 13. These data were then queried and analysed, the results of which are provided in Chapter 4. The results from this aspect of the work are supplemented with qualitative (bottom-up) observations. This analysis is conducted using content analysis techniques, as outlined in the next section (Section 3.4.5).

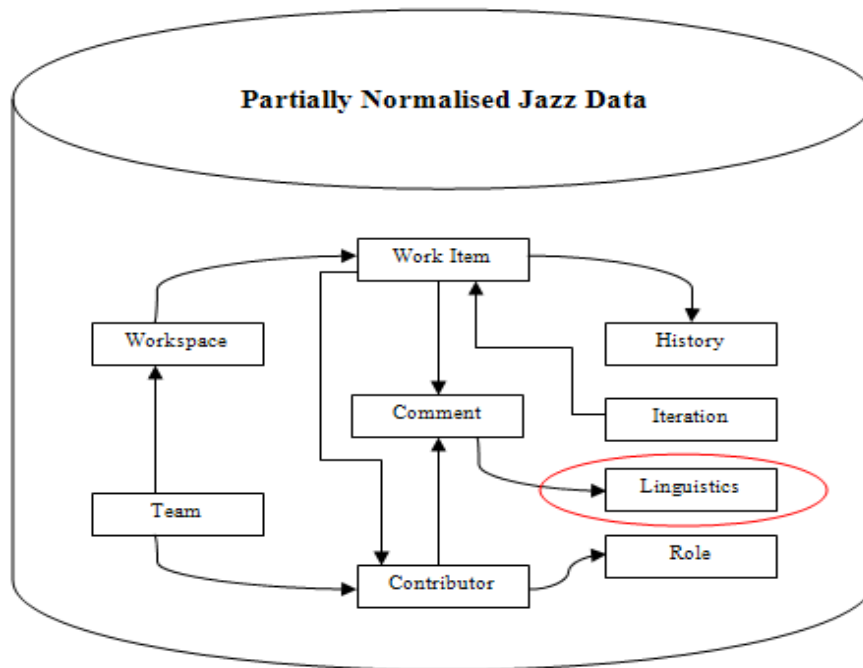


Figure 13. Extended database model after Linguistic processing

3.4.5 Data Analysis (Technique 3) - Content Analysis

Content Analysis (CA) is a technique commonly used by social scientists to study the content of recorded human communication (Babbie, 2004). From its early adoption, CA has also been used in the humanities to study the meaning, authenticity or authorship of texts. Therefore, in this section CA is investigated under the more established perspectives of authors from the humanities and social science disciplines.

Holsti (1969) explained that CA may be used to describe trends in communication content, relate known characteristics to a source, and compare content to standards. Fundamentally, as Harold Laswell, a popular communication theorist, outlined, content analysts use communication in seeking answers to questions such as “who says what, to whom, why, to what extent and with what effect”. This technique was developed by Alfred Lindesmith in 1931 to provide an alternative view for exploring, evaluating and analysing meaning, besides testing hypotheses, and was later supported by Glaser and

Strauss in the 1960s in their variant of this methodology known as Grounded Theory (Glaser & Strauss, 1967).

Content analysts are able to systematically and objectively study and identify properties in large amounts of information (Berelson, 1952). Krippendorff (2004) suggests that a valid and robust CA must consider six questions:

1. Which data are analysed?
2. How are the data defined?
3. What is the population from which the data are drawn?
4. What is the context relative to which the data are analysed?
5. What are the boundaries of the analysis?
6. What is the target of the inferences?

These questions, along with consideration of how CA has been used in SE and IS research, and the use of CA in this study, are addressed in the following eight subsections (Subsections 3.4.5.1 to 3.4.5.8). Firstly, the different forms of CA are explored in Section 3.4.5.1 to inform the selection of a specific approach for use in this study. The research is then informed by an examination of the theories around the CA process, including reliability and validity issues in CA (Section 3.4.5.2), the process for creating reliable and valid protocols (Section 3.4.5.3), selecting the unit of analysis (Section 3.4.5.4), the use of CA tools (Section 3.4.5.5) and ethical requirements in CA (Section 3.4.5.6). A brief survey of SE and IS studies that have employed CA techniques is presented in Section 3.4.5.7 to demonstrate the appropriateness of these techniques for studying communication in the way it is approached during this study. Finally, this section closes with a discussion of the procedures that were implemented during the CA phase of this work in Section 3.4.5.8.

3.4.5.1 Forms of Content Analysis

Quantitative content analysis considers keyword frequencies, word counts, message length and similar attributes in a descriptive way (Bullen, 1998). The fundamental assumption of this approach is grounded in the notion that frequently used words and phrases are often the most important and relevant to the specific scenario. Of course a key concept that occurs once may also be more important than a mundane issue that is commonly addressed. Quantitative content analysis may also be extended to include a

more linguistic, qualitative approach where words are assessed to explain context. This approach is often called Qualitative content analysis, where researchers aim to assess communicators' intentions and the implications of these intentions on some process or construct. The rationale for employing this approach is rooted in the notion that human behaviour has functions which may be revealed purposely or subconsciously while communicating (Bales, 1950a), regardless of the settings.

Under the Qualitative content analysis umbrella, analysis techniques range from intuitive and interpretive analyses to the more strict examination of data (Rosengren, 1981). The specific approach chosen for implementation is often aligned with the researcher's theoretical stance and the nature of the phenomenon of interest (Weber, 1990). Hsieh & Shannon (2005) classify these approaches as conventional, directed and summative. The conventional form of CA is used to describe phenomena where existing theories are limited, thus, the approach is aimed at theory building. As with the implementation of the grounded theory method, researchers employing this approach generally start the process of data analysis by inductively examining the data, allowing meaning to flow from the data as against approaching data analysis with any preconceptions (Mayring, 2000). Directed CA is used when there is scope to extend or complement existing theories around a phenomenon (Hsieh & Shannon, 2005). The directed content analyst approaches the data analysis process using existing theories to identify key concepts (and definitions) as initial coding categories. Should these prove inadequate during data analysis, new categories and subcategories are created to extend those found in previous theories to identify new meaning present in data. In directed CA it is also quite common for the data to drive the creation of new coding schemes altogether (as in the conventional form of CA above) (Hsieh & Shannon, 2005). Summative CA is also informed by previous theories and looks for evidence of predefined words or phrases to explore usage. This approach does not attempt to create new coding categories, and most often is used to illuminate or interpret the context of specific language usage (Hsieh & Shannon, 2005).

The use of a prior classification scheme to measure communication in the directed and summative forms of CA enables communication to be segmented into various units of analysis, and each unit can then be placed into a category, before the numbers of items belonging to each category are tabulated. These categories are then verified when related researchers examine this communication with an aim of having comparable interpretation of the communication data, as a means of increasing validity and

reliability (Weber, 1990). Sometimes one unit may be classified in multiple categories. Another method for categorising communication data is having participants code their own contributions, by providing a limited set of options (Ravenscroft & Pilkington, 2000). However, participants' lack of awareness of the CA method and the issues outlined above by Krippendorff (2004), in addition to the classification scheme providing limited options for coding, would likely result in incorrect categorisations – in the process affecting the work's reliability and validity. This issue is further discussed in the subsequent section (Section 3.4.5.2).

3.4.5.2 Reliability and Validity Issues in Content Analysis

Reliability and validity issues (see Neuendorf (2002) for reliability and validity discussions) have been a source of critical debate since the early uses of CA (Mowrer, 1996; Rourke & Anderson, 2004). In particular, while reporting frequencies, message lengths and message types (e.g., whether communication is a question or an explanation statement) is quite straightforward, and findings of studies examining these variables may not be challenged in terms of their quantitative representations, issues of validity surface when content analysts attempt to interpret psychological and mental constructs such as knowledge transformation, critical thinking or other social processes without considering or employing techniques developed by cognitive psychologists. For example, the measurement of students' cognitive ability by counting the number of propositions that followed an earlier statement (Henri & Kaye, 1992) may be criticised as invalid because of the nature of what is being inferred. The basis of such criticisms may be embedded in the fact that cognitive or meta-cognitive abilities are mental processes which may not necessarily exist in, or be evident in, written text (Bereiter & Scardamalia, 1987). For example: someone may transcribe memorised information without engaging any of their problem solving or knowledge transformation faculties while under observation for problem solving and knowledge transformation competencies, but nevertheless these occurrences may be recorded as valid evidence. Thus, content analysts are encouraged to take care in developing theoretically valid protocols in assessing the characteristics of the communication under observation when their aim is to make inferences (De Wever, Schellens, Valcke, & Van Keer, 2006). This issue is therefore examined in the next section (Section 3.4.5.3).

3.4.5.3 Creating a Reliable and Valid Protocol

For directed and summative forms of CA, developing a valid protocol (or classification scheme) involves identifying the purpose of the coding, identifying behaviours that represent the construct of interest, reviewing the categories and indicators, holding preliminary try-outs and developing guidelines for administering, scoring, and interpreting the coding scheme (Rourke & Anderson, 2004) (see Figure 14 below for illustration). Data may be coded to investigate many areas of communication, including the interactive, participative or social nature of communication. In this regard, nominal scales may be used to represent individual categories (e.g., the Gunawardena, Lowe, & Anderson (1997) scheme for coding social knowledge construction). After the coding scheme is created, the behaviours representing the construct are identified. This process is observational and aims to validate that the coding instrument considers only representative behaviours of the construct. For instance, in studying participation, ‘an individual commenting on a topic would denote a single representative behaviour’. However, exclusively studying individual participation in this form may not be particularly meaningful; thus, this may be combined with other variables, such as achievement (Richardson & Swan, 2003) or satisfaction (Gunawardena & Zittle, 1997).

Construct validity is critical when considering that behaviours should be representative. Literature reviews, the critical incident technique (CIT) and conventional forms of CA (see Section 3.4.5.1) are methods often used to identify behaviours that represent constructs (Rourke & Anderson, 2004). Categories of the indicators are then reviewed once representative behaviours are identified. Normally, this process is facilitated by domain experts who verify the provisional coding categories. Once the protocol is verified, the next step is for the content analyst(s) (ideally plural) to hold preliminary try-outs, and to check the instrument’s internal reliability (by assessing intra-rater agreement). Intra-rater agreement refers to the correlation of the results obtained by the same coder in the same study, while inter-rater agreement refers to the correlation of results obtained by multiple coders in the same study (Rourke & Anderson, 2004). This stage allows the researcher(s) to assess the coding scheme for shortcomings, in the process removing unwanted indicators or rewording those that are not worded adequately, and so on. In fact, the most valid coding schemes are those that have been used in multiple and distinct studies, especially if such schemes possess high inter-rater agreements (Gall, Borg, & Gall, 1996; Rourke & Anderson, 2004).

The last step in the protocol development process is the provision of guidelines for administering, scoring, and interpreting the coding scheme. This would normally comprise procedures created from the work conducted in the previous steps, and such guidelines would include training procedures for coders as well as sample coded scripts (see (Jonassen & Kwon, 2001) for further details). As in the try-out stage outlined previously, this step also seeks to assess the instrument's internal reliability.

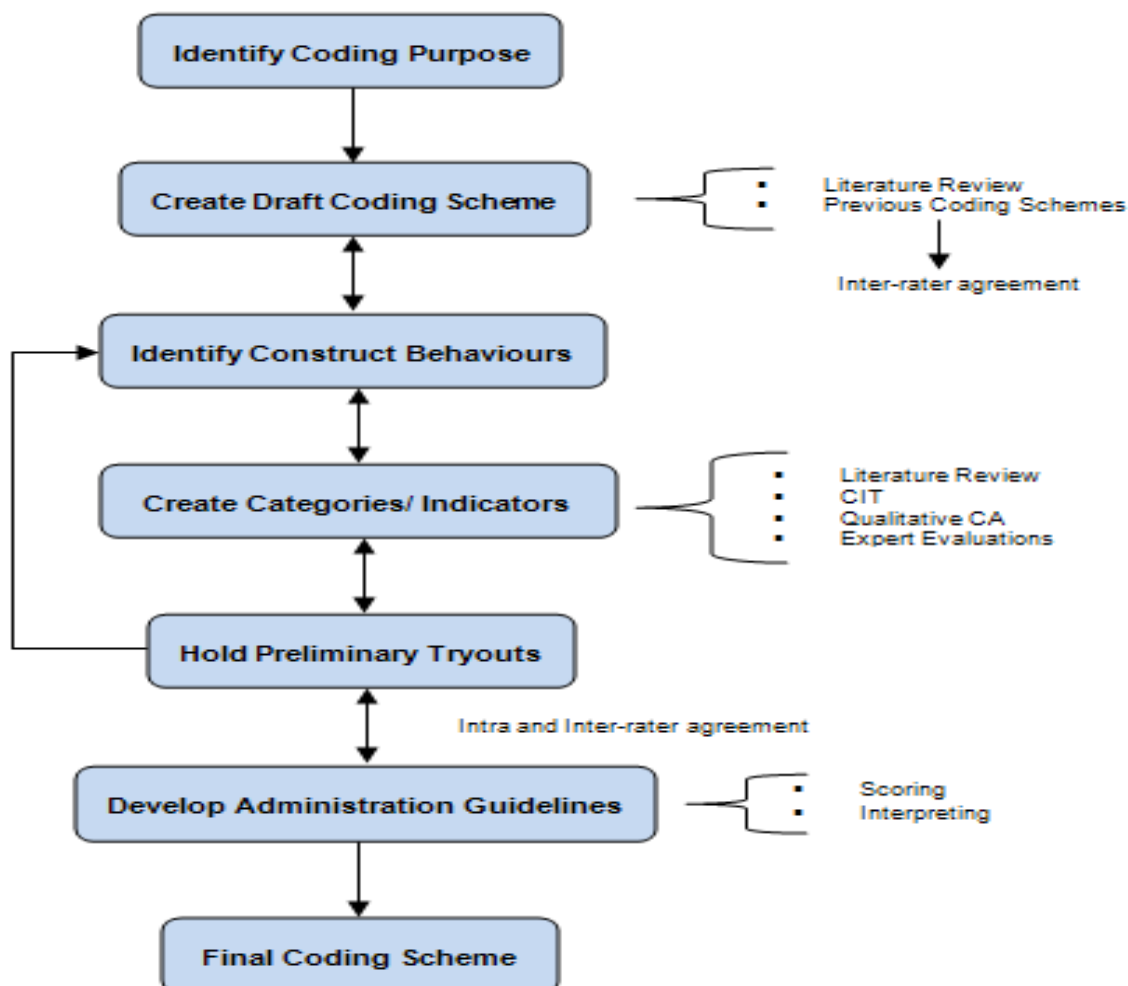


Figure 14. The Content Analysis protocol development process

Additional empirical evidence gathered from interviews or questionnaires administered to those being studied, or participants from a similarly representative population (Rourke & Anderson, 2002), may be used to support the validity of the coding instrument and the inferences made as a result of its use. Such an approach is classified as correlation analysis (Rourke & Anderson, 2004). In considering this theme, Messick (1989) also outlined two additional types of investigation that may support this activity – examination of group differences and experimental interventions. In the group difference investigation, the instrument is administered to more than one group of communicators, with at least one of the groups being less representative of the typical

behaviour under investigation. The instrument under consideration should distinguish between these groups, providing support for its validity. In the experimental scenario, content analysts aim to modify the behaviour of the domain under consideration to verify that the instrument detects this change.

This work uses multiple analysis techniques, including SNA, linguistic analysis and directed CA. The linguistic analysis results were correlated with the CA results in a form of experimental intervention. Additionally, the protocol that was used in this work was validated previously, and also tested in this work for accuracy, precision and objectivity. Further details on this process are provided in Section 3.4.5.8.

The themes and behaviours are frequently studied from communication that has been segmented into various units of analysis. The selection of the study's unit(s) of analysis can in itself provide some specific challenges. These challenges are considered in detail in Section 3.4.5.4.

3.4.5.4 Selecting a Unit of Analysis

Another area briefly highlighted in Section 3.4.5.1 that warrants further consideration in this CA discussion is the approaches used for segmenting communication into different categories (based on the unit of analysis). The goal of segmenting a communication artefact transcript into discrete units is to maximise the likelihood of repeatability among coders categorising constructs so that objective patterns are revealed (De Wever et al., 2006). Researchers recognising single words, sentences, messages or paragraphs are said to be utilising 'syntactical or fixed units' to identify constructs, whereas those distinguishing communication by 'themes' or 'ideas' (units of meaning) are said to be employing dynamic units (Rourke, Anderson, Garrison, & Archer, 2001). The suitability of employing these approaches depends on the characteristics of the communication under consideration (Schrire, 2006). For instance, in informal communication where ellipses and other symbols are used to denote mood (e.g., Was about to use that one... what do you think... anyway...lol) decoding transcripts into fixed units could be challenging and might often require an additional subjective step of interpreting the message prior to its classification (Blanchette, 1999). When the paragraph is used as the unit of analysis, multiple variables of interest may exist in one unit, affecting the reliability of the study. Similarly, utilising the more dynamic unit of meaning could be very challenging when multiple researchers are coding subjective

constructs such as critical thinking or knowledge transformation; see Henri & Kaye (1992) for example.

Thus, caution must be exercised when selecting a unit of analysis, and typically, the nature of the communication content should drive the unit of analysis that is selected (Schrire, 2006) (Section 3.4.5.8 outline the specific approach that is used during this work). These segmented communications are often managed through the use of software tools; these are examined next (refer to Section 3.4.5.5).

3.4.5.5 Content Analysis Tools

There are many software packages that provide support for both quantitative and qualitative forms of CA (for a list of those available see <http://www.content-analysis.de/software>). Those supporting quantitative CA include Concordance⁵ and Diction⁶, while qualitative packages include ATLAS.ti⁷, HyperResearch⁸ and NVivo⁹. These software packages support coding and analysis, utilising frequency counts and pattern matching, and output reports with the numbers of messages and coder identification, among other information. Statistical analysis packages such as SPSS¹⁰ may also be used to support (further) quantitative analyses. Before such tools are used, however, it is critical for researchers to gain the appropriate permissions to study communication artefacts; this issue is considered next (see Section 3.4.5.6).

3.4.5.6 Ethical Requirements of Content Analysts

Ethical approval and consent issues are relevant to all research that employs human subjects. Content analysis studies bear no exception to this rule. Permission is often required from participants involved in research where transcripts are analysed if the private information of participants is identifiable or the researcher was previously involved in interaction with the participants, and so, may be able to recall specific scenarios, in the process, being able to identify participants. These guidelines have been considered in the study of many software engineering related issues, as examined in Section 3.4.5.7.

⁵ <http://www.concordancesoftware.co.uk/>

⁶ <http://www.dictionsoftware.com/>

⁷ <http://www.atlasti.com/index.html>

⁸ <http://www.researchware.com/products/hyperresearch.html>

⁹ http://www.qsrinternational.com/products_nvivo.aspx

¹⁰ <http://www-01.ibm.com/software/analytics/spss/>

3.4.5.7 Content Analysis in SE and IS Research

A review of the literature has shown that researchers in the SE and IS discipline areas have employed the CA method to examine a range of issues. A selection of the more relevant studies is considered here. Content Analysis was used by Sheetz, Henderson, & Wallace (2009) to investigate software managers' and developers' perceptions of the various methods used for software estimation. Hall, Wilson, Rainer, & Jagielska (2007) employed CA to analyse communication among participants in a large software development project. The methods used for task assignment in OSS projects were examined using CA by Crowston, Li, Wei, Eseryel, & Howison (2007). Zannier, Chiasson, & Maurer (2007) employed CA to investigate the ways in which software designers make their design decisions. Barcellini et al. (2008) utilised CA to model team dynamics during the OSS design process. Ryan & O'Connor (2009) studied the effects of tacit knowledge, explicit job knowledge and social interaction on team performance using CA. Quantitative aspects of the CA technique were also employed by Gallivan (2001) to investigate the variables most relevant for effective performance of OSS projects. The applications of CA techniques in these works, although variable in their specific nature, have helped these researchers to provide valuable insights into the software development process. This approach is similarly adopted in this work, as outlined in the following section (Section 3.4.5.8).

3.4.5.8 Use of Content Analysis in this Study

Content analysis techniques are used during this research to study core developers' enacted roles, their knowledge sharing behaviours, and the way core developers become hubs in their team. These issues are studied directly from core developers' messages. Of the different variants of the CA method introduced in Section 3.4.5.1, it may be apparent that directed CA is most suitable for undertaking the exploratory aspects of this work. This study examines the aforementioned issues (refer to Section 3.4) by studying developers' interactions. Interactions have been studied repeatedly using CA techniques in the education and group work domains, and thus, pointers in these works provided utility for this research project.

For instance, Henri & Kaye (1992) and Hara, Bonk, & Angeli (2000) used CA to study interaction using computer mediated communication (CMC). Newman, Webb, & Cochrane (1995) studied group learning and critical thinking using a similar qualitative approach. Zhu (1996) considered team members' interactions during group problem

solving. Gunawardena et al. (1997) provided a model for studying social construction of knowledge using a variant of the CA technique (grounded theory). Fahy, Crawford, & Ally (2001) utilised Zhu (1996)'s model towards developing an instrument to guide qualitative examination of communication data for evidence and results of interaction. Aviv et al. (2003) and De Laat et al. (2007) went one step further while studying communication to employ both CA and SNA in the examination of knowledge construction processes and interaction and knowledge patterns in learning communities.

The studies of Henri & Kaye (1992) and Zhu (1996) are particularly applicable to the work undertaken in this research because of their treatment of interaction and knowledge sharing, key aspects under consideration in this research project. In fact, the Henri & Kaye (1992) study on computer supported learning (CSL) is one of the pioneering work on teams' interaction; it provides many interesting findings, and is one of the most cited CA instruments used to study interaction (De Wever et al., 2006; Erlin et al., 2008; Manca, Delfino, & Mazzoni, 2009; Naidu & Jarvela, 2006). The coding instrument of Henri & Kaye (1992) is grounded in the cognitive approach to learning and interactivity. Interactivity, as used in Henri's model, was premised on the Bretz (1983) three-stage theory of interactivity – where an individual provides initial communication of information, someone responds to this information and a follow up communication to the first communication is then transmitted. Henri's coding instrument was created to observe five areas of interactivity: participation, social, interaction, cognitive and meta-cognitive communication. Interaction, the theme under consideration in this research, was measured by the number of messages communicated and the activeness of the participants (e.g., communication directly/indirectly related to content). The dynamic unit of meaning was utilized by Henri & Kaye (1992), an approach that allowed them to validate precisely the way each individual communicated when contributing. Henri's instrument has been tested for inter-rater reliability by many researchers; for example, Lockhorst, Admiraal, Pilot, & Veen (2003) reported a Cohen's kappa of 0.73 agreement among coders in assessing the nature of communication content, while Hara et al. (2000) recorded 0.78 agreement for the "task/off task" (social) scale. These figures represent good to excellent agreement beyond chance (Banerjee, Capozzoli, McSweeney, & Sinha, 1999).

Another widely cited study (noted earlier) that previously utilised computer mediated communication in the education and group work disciplines, and that is of relevance to this work is that of Zhu (1996). Zhu grounded her protocol in the group interaction

theory of Hatano & Inagaki (1991) and adopted Graesser & Person (1994) approach to question analysis, where social interaction was considered to be multi-dimensional. According to Zhu (1996), social interaction may be classified as vertical interaction or horizontal interaction. Vertical interaction is characterised by communication where group members seek answers or solutions to problems from capable members as against these individuals being willing to make personal contributions to the knowledge construction process (e.g., information-seeking questions posed because of some previously incomplete communication or a feeling of information deficit). Such individuals, while not seen as active participants in the knowledge construction process, are useful in groups, especially where other members may have the same information deficiencies but are unwilling to seek that information. In addition, it may also be advanced that those responding to such questions are given an opportunity to develop their own understandings towards knowledge maturity.

Horizontal interaction, on the other hand, involves the strong assertion of ideas, which is perceived as necessary in environments where little authoritative feedback will be forthcoming. Zhu (1996) further divided this horizontal category into a number of sub-categories. In particular, Category II labelled as ‘assertions’ are aimed at seeking opinions or starting a dialogue. Additional categories of this instrument (considered under the horizontal classification) are ‘answers’, ‘information sharing’, ‘discussion’, ‘comment’, ‘reflection’ and ‘scaffolding’. Answers are specific responses to vertical interaction (questions), whereas, information sharing is of a more general nature. Communications that are intended to share ideas are discussions, and comments are those communications that are non-probing or judgmental statements. Reflective communication was described by Zhu to be evaluative, linking previous communications and adjusting previous viewpoints, whereas scaffolding communications are those providing guidance or suggestions related to specific tasks. Zhu’s protocol has also been tested for inter-rater reliability; for instance, Fahy et al. (2001) reported 86% reliability using Zhu’s protocol in coding students’ and instructor transcripts which was verified by three independent coders.

The conceptual framework upon which the two coding instruments just outlined were built provides a strong basis for understanding how individuals collaborate beyond the surface of communication or the regularly reported quantitative indicators of interaction (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009). While these coding instruments have been mostly employed previously in educational

and group work settings, there is relevance in applying these frameworks in the SE and IS disciplines to examine goal-directed group work and problem solving and to observe actual software process communication in detail. In particular, theories such as those of Vygotsky (1978) that emphasise the role of the environment on an individual actions could be verified by considering different phases of the group process, using both quantitative and qualitative methods to assess interaction and knowledge construction and maturity among group members, thus providing an extension to the body of knowledge regarding group work, team communication and the structure of interaction in these domains. Additionally, as noted in Chapter 2, findings from such observation are likely to offer support for those provided by SNA studies (Erlin et al., 2008) that consider team members' interactions in purely quantitative ways (Abreu & Premraj, 2009; Bird et al., 2006a; Nguyen, Wolf, et al., 2008), extending these to provide detailed explanations of the *nature* of interactions in social networks and how specific patterns of interaction contribute to group task performance.

Accordingly, a protocol was created following the processes outlined in Figure 14 and all of the messages contributed by three teams are studied (1561 messages from P1, P7 and P8 – see Table 2) using directed CA. Artefacts from these three teams are deliberately selected as they comprised development efforts aimed at different types of software features (P1 = User Experience tasks, P7 = Coding tasks, and P8 = Project Management tasks). Thus, it is anticipated that potential variations in interaction among those involved in different forms of task would also be revealed by studying the artefacts associated with these teams. The protocol, shown in Table 6, is a hybrid classification scheme adapted from Henri & Kaye (1992) and Zhu (1996). As noted above in Section 3.4.5.1, use of a directed CA approach is appropriate when there is scope to extend or complement existing theories around a phenomenon (Hsieh & Shannon, 2005), and so the directed CA approach is suitable for exploring the issues that were identified in Chapter 2, and particularly for understanding how and why core developers contribute to agile globally distributed team dynamics. This contextual analysis approach also offers avenues for triangulating the findings that are revealed during the SNA and linguistic analysis phases (refer to Section 3.4.3 and Section 3.4.4).

In Section 3.4.5.3, it is outlined that the directed content analyst approaches the data analysis process using existing theories to identify key concepts and definitions as initial coding categories. As noted earlier, existing theories (and protocols previously tested for studying interaction) are used for understanding practitioners' enacted roles

(refer to Section 2.7.1), and the way knowledge sharing behaviours (refer to Section 2.7.2) are expressed during textual interaction.

Henri & Kaye (1992) and Zhu (1996) protocols were used to inform the creation of the initial coding categories (see scales 1 to 8 in Table 6 shown in black font face). Preliminary coding try-outs were then conducted, following the process depicted in Figure 15. Initially, a small sample of messages was selected for coding by the main researcher, and these were coded according to scales 1 to 8 in Table 6. Communications not captured under these scales (1 to 8) were coded as “Not Coded” (scale 9 of the initial protocol) – see Figure 15. During this process it was noted that the initial protocol (scales 1 to 8 in Table 6) was inadequate for studying Jazz practitioners’ utterances, as a large number of codes were being recorded under the “Not Coded” scale.

This issue is regularly encountered during the CA process (Hsieh & Shannon, 2005; Rourke & Anderson, 2004). In Section 3.4.5.1 it is explained that during the directed CA process, should existing theories prove insufficient to capture all relevant insights during preliminary data analysis, new categories and subcategories should be created (Hsieh & Shannon, 2005). Therefore, in this study’s context, the main researcher, the primary supervisor and two other trained coders formally categorized 5% of the three teams’ communications (80 messages) in a second coding phase. Coders were provided with guidelines for administering, scoring, and interpreting the coding scheme; including examples of messages that were coded under each category (refer to Figure 15 for the initial coding process).

During this second phase of coding it was also noted that practitioners in Jazz communicated multiple ideas in their messages. Thus, messages were segmented using the sentence as the unit of analysis. The initial protocol was extended to include new scales directly from the pilot Jazz data (see scales 9 to 13 in Table 6, shown with blue font face). This extended protocol was derived by consensus among the entire team of coders (the main researcher, the primary supervisor and two other trained coders). Thereafter, all the messages were re-coded by the main researcher and the two trained coders. Duplicate codes were assigned to utterances that demonstrated multiple forms of interaction, and all coding differences were discussed and resolved by consensus (see Section 4.2 for further details). Through the use of Holsti’s coefficient of reliability measurement (C.R) an 81% inter-rater agreement level between the three coders was recorded (Holsti, 1969). This represents excellent agreement between the coders.

Table 6. Coding categories for measuring interaction

Scale	Category	Characteristics and Example
1	Type I Question	Ask for information or requesting an answer – “Where should I start looking for the bug?”
2	Type II Question	Inquire, start a dialogue - “Shall we integrate the new feature into the current iteration, given the conflicts that were reported when we attempted same last week?”
3	Answer	Provide answer for information seeking questions - “The bug was noticed after integrating code change 305, you should start debugging here.”
4	Information sharing	Share information – “Just for your information, we successfully integrated change 305 last evening.”
5	Discussion	Elaborate, exchange, and express ideas or thoughts – “What was most intriguing about solving this bug is not how bugs may exist within code that went through rigorous testing... but how refactoring reveals bugs even though no functional changes are made.”
6	Comment	Judgemental – “I disagree that refactoring may be considered the ultimate test of code quality.”
7	Reflection	Evaluation, self-appraisal of experience – “I found solving the problems in change 305 to be exhausting, but I learnt a few techniques that should be useful in the future.”
8	Scaffolding	Provide guidance and suggestions to others – “Let’s document the procedures that were involved in solving this problem 305, it may be quite useful for new members joining the team in the future.”
9	Instruction/ Command	Directive – “Solve task 234 in this iteration, there is quite a bit planned for the next.”
10	Gratitude/ Praise	Thankful or offering commendation – “Thanks for your suggestions, your advice actually worked.”
11	Off task	Communication not related to solving the task under consideration – “How was your weekend?”
12	Apology	Expressing regret or remorse – “I am very sorry for the oversight, and I am quite unhappy with the failure this has caused.”
13	Not Coded	Communication that does not fit codes 1 to 12.

Given the multi-method approach that is utilised during this work (refer to Section 3.4), there existed many avenues for correlation analysis to further verify the validity of the final instrument that was created for analysing practitioners’ interactions in Table 6. These results are reported in Section 4.3.5. Additionally, ethical issues considered under Section 3.4.5.6 were not of direct concern in this work as the private information of the practitioners that are selected for observation was not identifiable in the repository, nor was the researcher previously involved in interactions with the participants. During the coding process, messages were ordered around each work item (task) according to the date they were contributed. This allowed the coders to readily identify the development of the communication threads during practitioners’ exchanges. Once coding was completed, the codes were entered into an extended version of the Microsoft SQL relational database (as was done for the imported linguistic analysis data – refer to Section 3.4.4.2) through a simple Visual Basic user interface that was created to facilitate this data entry process (see Figure 16). Figure 17 provides an illustration of the extended database model.

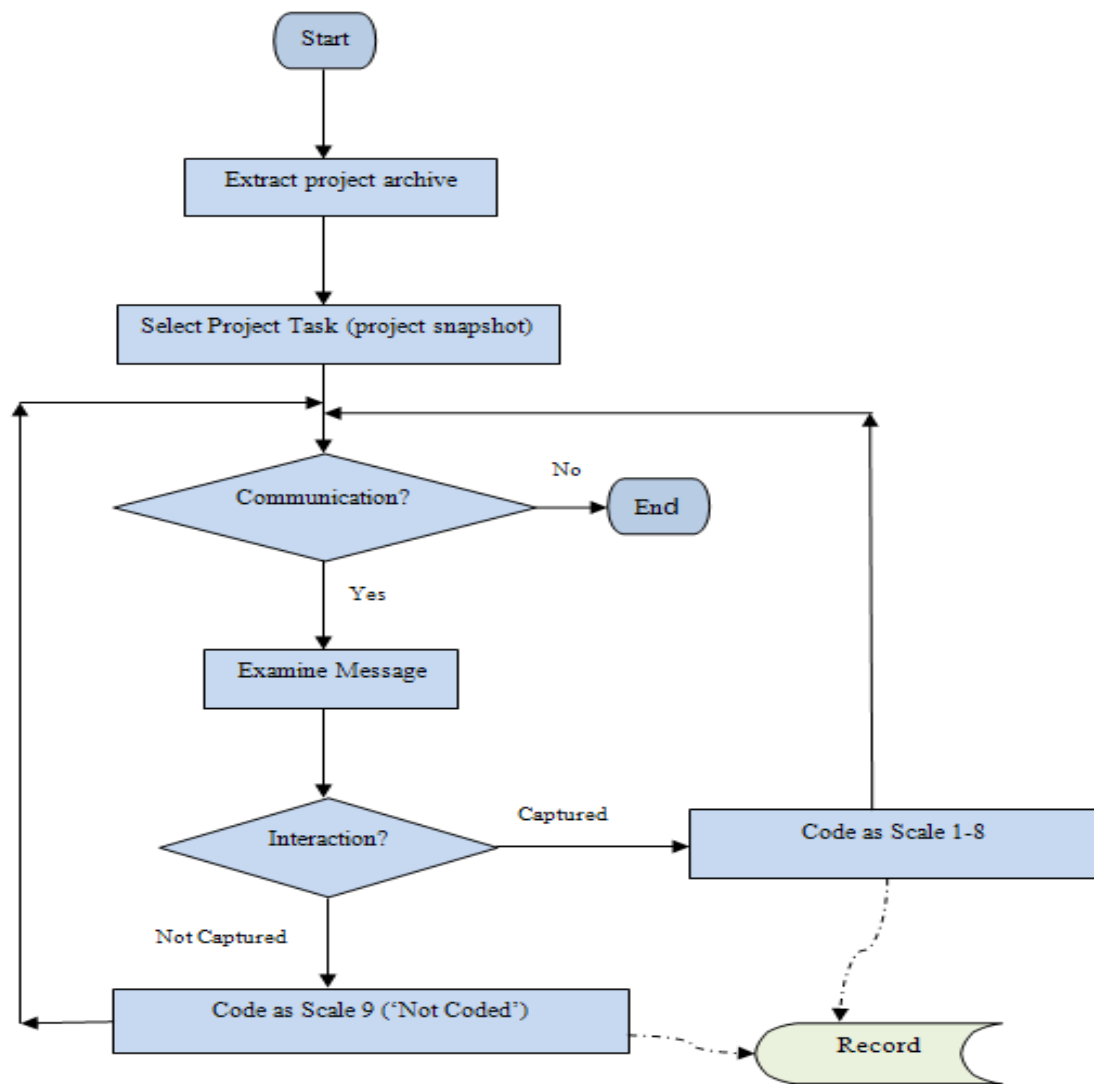


Figure 15. Initial coding process

Add Codes

Code_ID:

Comment_ID:

Code:

Edit Codes

Code_ID	Comment_ID	Code
1	15611	4
2	15611	8
3	15611	4
4	15612	4
5	15612	8

Figure 16. Simple visual basic interface that was created for entering codes into the Microsoft SQL database

The data analysis techniques described in Section 3.4.3, Section 3.4.4 and Section 3.4.5 facilitate the delivery of multiple stages of results that are aimed at answering the research questions posed in Figure 4 (refer to Chapter 4). These answers support the development of this study's theoretical contributions. Section 3.5 considers this issue at length, including discussions around the process of theorising, with an emphasis on

theory generation in SE research and details concerning the nature of the theoretical contributions that are provided in this work.

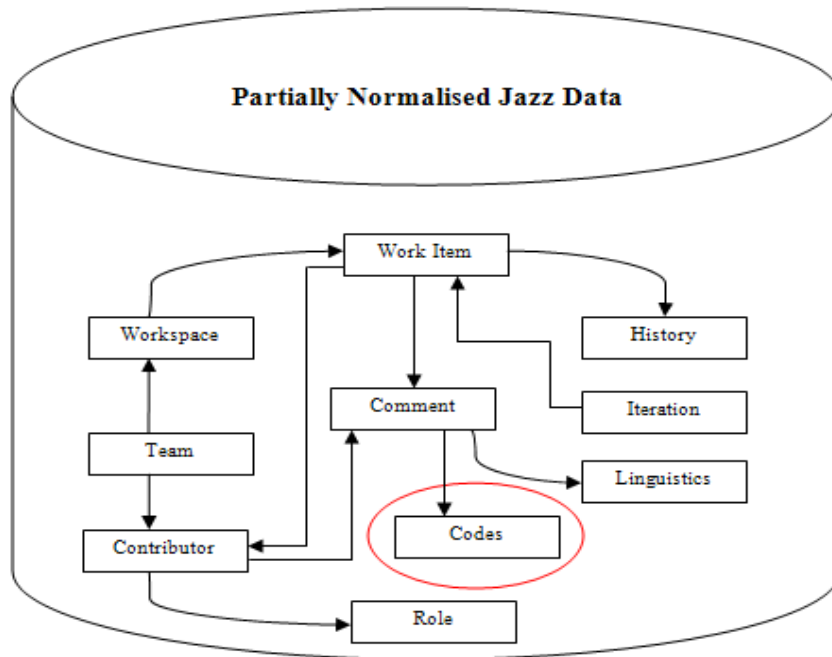


Figure 17. Extended database model after Coding process

3.5 Process of Theorizing

Theory takes different shapes and forms depending on the research discipline and the domain of interest. For instance, in the physical or natural sciences, theories are seen to provide explanations or predictions for specific events and patterns (Propper, 2005, p. 59), whereas in the social sciences, theories tend to identify relationships among constructs, with a requirement that these relationships should be testable (Doty & Glick, 1994, p. 233). Similarly, the meaning of theory tends to shift from discipline to discipline; for instance, theory defined in the mathematical sciences is quite different to that in the design or social sciences. Theory may also be viewed from a philosophical position; while logical positivism emphasises verifiable or factual viewpoints, interpretivists view theories as being derived from actors' accounts of specific contextual situations (Godfrey-Smith, 2003). A broader definition of the word theory taken from the Oxford English Dictionary (OED) is "a supposition or a system of ideas intended to explain something, especially one based on general principles independent of the thing to be explained" (OED-Online). Taken further, theories are seen to offer universal conceptual frameworks from which knowledge may mature, or actual understandings for what is often commonly (and sometimes speculatively) assumed (Lewin, 1945).

While there is an abundance of studies around the process of theorising in most of the established disciplines (e.g., the natural sciences (Kuhn, 1970) and psychology and social sciences (Davis, 1971; Dubin, 1978)), this subject has received far less attention in the software engineering literature (Johnson, Ekstedt, & Jacobson, 2012). In fact, the very applicability of the theory generation process in the applied disciplines has been questioned (Lynham, 2002). Thus, even when solid propositions that may form the basis for theories are provided (e.g., those in the SWEBOK (SWEBOK, 2004)), these are often not promoted as initial theories (note: Johnson et al. (2012) argue that the SWEBOK may actually serve as a theory that describes the software engineering process). Some argue that this issue is tenuous at best, and has the potential for the software engineering discipline to remain a “trial and error” paradigm (Johnson et al., 2012).

Moreover, the relevance of theory use and theory generation in the SE literature has been echoed by some of those conducting research in the discipline. Some argue that SE research outputs are not useful if there is no theoretical basis for conducting the particular study (Hannay, Sjoberg, & Dyba, 2007); this is likely to be true if the intent is to solve a general class of problems rather than a specific problem instance. Additionally, it has been asserted that the absence of theories is detrimental in terms of inspiring research enquiries, and that theory-led empirical studies may improve the state of both SE research and practice, by deriving more rigorous inspection of methods (Johnson et al., 2012). Theory-led enquiries may also result in unintended benefits for other disciplines (Hannay et al., 2007).

Accordingly, given the close alignment between the SE and IS disciplines (e.g., see Morrison & George (1995) early review of the IS literature which found that 45% of the work in this body belonged to the software engineering discipline), those theorising in SE research often adopt Gregor’s IS classification scheme (Gregor, 2006; Hannay et al., 2007; Johnson et al., 2012). Gregor (2006) contends that IS theories tend to take three forms depending on the phenomenon of interest – theories aiming to generalise local observations into abstract knowledge, theories aimed at representing causality (cause - effect) and theories that explain or predict events. Consequently, Gregor (2006) classified these according to five different types: 1. Analysis, 2. Explanation, 3. Prediction, 4. Explanation and prediction, and 5. Design and action. Type 1 theories (Analyses) typically include descriptions, conceptualisations, taxonomies and classification schema, and say “what is”, e.g., the SWEBOK (SWEBOK, 2004). These

forms of theories are often relevant when very little is known about the phenomenon of interest (Miles & Huberman, 1994).

Type 2 theories (Explanations) are aimed at providing understandings and answers for questions related to how and why specific events occur or how and why certain patterns exist, e.g., Structuration theory (Giddens, 1984). Gregor (2006) explained that case studies typically provide such theories. Type 3 theories (Predictions) provide predictions for what will happen, but rarely explain why the specific pattern exists, e.g., the COCOMO model (Boehm et al., 1995). Research studies providing prediction theories typically involve statistical analysis and quantitative methods such as data mining, correlation testing and regression analysis. Type 4 theories (Explanations and predictions) aim at explaining and predicting events, answering questions such as how, why, when and what will be, e.g., the Technology Acceptance Model (TAM) (Davis, Bagozzi, & Warshaw, 1989). These theories are aligned with the views of both natural and social scientists, where studies are aimed at building and testing theories.

Finally, Type 5 theories (Design and action) specify how to do something. This form of theory is also referred to as software engineering or the system development approach (Gregor, 2006). Hevner, March, Park, & Ram (2004) referred to this approach in a broader context of design science. Theories belonging to this categorisation are generally presented in the form of software artefacts (Hevner et al., 2004), although, models, evaluations and metrics are also commonly delivered as “Design and action” theories (Gregor, 2006), e.g., the ASRM tool supporting team composition (Licorish et al., 2009b). Gregor (2006) noted that multiple theories may also be provided in a single body of work due to the way many of these theories are related, e.g., the development of most Explanation theories (categorised as Type 2 under Gregor (2006) model) starts with an Analysis (categorised as Type 1 under Gregor (2006) classification scheme).

Empirical software engineering studies (as is conducted in this research) are typically aimed at explaining passing observations, to provide understandings for how and why phenomena occur (Hannay et al., 2007). Particularly, such theories are helpful for understanding the conditions under which specific development approaches and practices are most useful during the software development process. Theories generated during such investigations are typically categorised under Gregor (2006) the Type 2 classification scheme (Explanations), and may be aimed at providing casual explanations (Hannay et al., 2007).

Although these forms of theories are indeed generated and extended in software engineering research, there is little explicit sharing of theories in the discipline. Hannay et al. (2007) systematic review covering a decade (between 1993 and 2002) of software engineering studies that were aimed at providing casual explanations shows that even when multiple software engineering studies are published on the same topic, different theoretical rationales were used. Given this state of affairs, some researchers have expressed doubt about the potential maturity of the body of work in the software engineering discipline (Johnson et al., 2012). Hannay et al. (2007) found that theories are used largely for rationalising the specific research approach, but that few software engineering studies attempted to extend these theories, or provide new ones altogether. Earlier work had noted that empirical support for prior claims is particularly rare in the empirical SE literature (Herbsleb & Mockus, 2003b), although, there is general recognition around the *need* for theories (noted earlier) (Hannay et al., 2007). Hannay et al. (2007) believes that the current state of play is driven by a lack of clear description of theories use and the theories created; accordingly, there is little follow up work.

There are some exceptions in the software engineering literature both in terms of theory use and theory generation however. For instance, Greenblatt & Waxman (1978) seminal work examined the ease of learning for SQL and QBE query languages and revealed that study subjects found QBE much easier to grasp than SQL. This work was later replicated by Boyle, Bury, & Evey (1983) whose findings contradicted Greenblatt & Waxman (1978); they discovered that SQL required less time to learn and study subjects preferred SQL over QBE. Yen & Scamell (1993)'s work was motivated by this disparity, and they took a slightly different approach to study this subject by considering equal treatment groups in a controlled laboratory experiment. Their findings show that study subjects' performance was higher for QBE than SQL in paper and pencil testing; however, there was no difference in study participants' performance for these language types in an online setting. Yen & Scamell (1993) also observed that the complexity of the query had an effect on users' performance. Multiple aspects of Yen & Scamell (1993) theory has since been verified by other works (De, Sinha, & Vessey, 2001; Groth, 2005).

Another example of the use and reuse of software engineering theories is demonstrated in the works of Lloyd & Jankowski (1999) and Berenbach & Borotto (2006). Lloyd & Jankowski (1999) employed cognitive information processing (CIP) and information theory (IT) in an experiment to study the clarity of data flow diagrams (DFD), and

revealed that the treatment group that was exposed to CIP and IT interventions was much faster at comprehending the DFDs than those that were in the non-CIP IT group. Berenbach & Borotto (2006) has utilised aspects of the Lloyd & Jankowski (1999) approach in real projects at Siemens Corporation in order to demonstrate their effectiveness, and have discovered promising results.

Further evidence of theory generation and use is demonstrated in Sauer, Jeffery, Land, & Yetton (2000), and subsequent works. Sauer et al. (2000) proposed the use of behavioural theory of group performance to explain the outcomes of software engineers' technical reviews. Rigby & Storey (2011) built on the Sauer et al. (2000) study and considered developers mechanisms and behaviours that facilitate peer review on OSS projects. Babar, Kitchenham, & Gorton (2006) had also previously used Sauer et al. (2000) work to motivate their own research that proposed a framework for distributed software architecture evaluation. Other works identified in Wieringa, Daneva, & Condori-Fernández (2011) have similarly used theories from other disciplines to motivate enquiries around software development/engineering issues.

This PhD follows a similar approach and grounds the issues under consideration in theories, in order to also provide insights that form the basis of a software engineering theory. This work presents an amalgamation of research efforts, some of which have been published as listed at the beginning of this thesis, and so provides initial theories for explaining (Gregor (2006) Type 2 classification scheme) a range team of issues (see Chapter 2). Given that previous work has studied software teams' communication patterns extensively (e.g., (Abreu & Premraj, 2009; Bird et al., 2006a; Cataldo & Ehrlich, 2012; Hinds & McGrath, 2006; Yu et al., 2011)), in addition to informing the latter stages of the analyses conducted in this research, the role of this aspect of the current work (see confirmatory research perspective in Table 7) is to seek confirmation for the collaboration patterns of successful globally distributed agile software teams (refer to Section 2.6 for additional details). Such a step is necessary, as oftentimes, such confirmations lead to stronger theories (Hannay et al., 2007). Sometimes the outcome of adaptations of other theories is beneficial beyond its intended purpose, and informs these host studies. Some also suggest that it is important for theories to emerge through iterative cycles of development as against the goal of delivering a grand theory (Pfleeger, 1999), an approach that is adopted in this work – the strategy being to conduct this preliminary analysis before deeper examinations of the attitudes and

behaviours of core developers. Such an iterative approach to theorising should lead to robust theories, which result after significant amounts of empirical work.

In addressing the second set of research issues that were outlined in Chapter 2, this work extends previous theories and provides understandings into the way core developers contribute to their teams' dynamics (refer to Table 7 for summary). While previous work has identified that a few software practitioners occupy the centre of their teams' knowledge processes (Bird et al., 2006a; Shihab et al., 2010), there was no previous attempt at understanding the reason for such a phenomenon (Licorish & MacDonell, 2013c). Questions related to the reasons for these members' extraordinary presence, and understanding the actual roles (both formal and informal) that core developers occupy in their teams, have not been answered. Additionally, there has been no attempt to reveal explanations for how developers become core become part of the core group. Furthermore, there still remain doubts regarding when these individuals are more or less likely to contribute the most to task performance and when their teams are most likely to benefit from their knowledge and experiences.

This first confirmatory analysis stage (also summarised in Table 7) provides explanations of successful globally distributed agile teams dynamics, and informs the subsequent investigations (conducted in two phases) centred around core developers (refer to Section 2.7). These latter investigations are largely exploratory (refer to Table 7) and are aimed at explaining why globally distributed software development teams exhibit centralised communication patterns, and the true nature of core developers' performance. Thus, from a theoretical perspective, this work delivers initial conjectures that may form the basis of explanation theories for understanding the collaboration patterns of successful globally distributed agile teams and how and why core developers contribute to globally distributed agile team dynamics (refer to Section 3.6 for a summary of this chapter, and an illustration of this work's methodological framework).

Table 7. Summary of research perspectives, questions and study techniques and measures

Research Perspective	Research Question(s)	Study Technique(s) and Measure(s)	Results
Largely Confirmatory (Phase 1)	RQ1. Do communication patterns change as the software project progress?	Aggregation of messages over project phases (frequency-based analysis)	Refer to Section 4.1.1
	RQ2. Is there equity in practitioners' contributions to their project?	SNA – Sociograms, In-degree, Density, Unique edges, Closeness	Refer to Section 4.1.2
	RQ3. Are active communicators more important to their teams' collaboration?	SNA – In-degree, Density, Closeness, Sociograms, Role details, Task performance (change logs)	Refer to Section 4.1.3
	RQ4. How are active communicators involved in task performance?		
	RQ5. Are practitioners' formal role assignments related to their involvement in project interactions and task performance?		
Largely Exploratory (Phase 2 and Phase 3)	RQ6. Do core developers' behaviours and attitudes differ from those of other software practitioners?	Linguistic Analysis – Behaviours and attitudes	Refer to Section 4.2.1
	RQ7. What are the core developers' enacted roles in their teams, and how are these roles occupied?	Directed Content Analysis – Enacted roles	Refer to Section 4.2.2
	RQ8. Do core developers' attitudes change as their project progresses?	Linguistic Analysis – Behaviour and attitudes (over project duration)	Refer to Section 4.3.1
	RQ9. How do core developers share knowledge over the course of their project?	Directed Content Analysis – Knowledge sharing, Becoming team hubs (over project duration)	Refer to Section 4.3.2
	RQ10. What initial team arrangements lead to developers becoming hubs in their teams?		
	RQ11. How do core developers contribute to task performance over their project?	Change Logs – Task performance	Refer to Section 4.3.3
	RQ12. Are core developers' contributions to task performance linked to their attitudes?	Linguistic Analysis and Change Logs – Attitudes and task performance	Refer to Section 4.3.4
	RQ13. Are core developers' contributions to task performance linked to their contribution of knowledge?	Directed Content Analysis and Change Logs – Knowledge sharing and task performance	Refer to Section 4.3.5

3.6 Chapter Summary and Methodological Framework

This chapter has described and justified the study's research methodology and design. In Section 3.1 it was shown that empirical studies using repository data have largely used frequency-based and mathematical analysis techniques to provide understandings of software engineering teams' communication processes. However, there are uncertainties around the suitability of these approaches for studying deeper psychological aspects of human-centric processes. Accordingly, techniques from the behavioural sciences and the organisational psychology domain have been recommended for supplementing the surface approaches to provide contextual understandings for teams' behavioural processes. The particular use of such deeper approaches is often tied to the nature of the research questions that are outlined and to

the researcher's theoretical stance. In particular, Section 3.2 explained that research aimed at testing or confirming theories generally conforms to a positivist framework, whereas studies aimed at providing understanding of new and untested phenomena are often exploratory in nature and conform to a more interpretivist approach.

The issues that are explored in this study demand the utilisation of techniques that are aligned with both positions (positivism and interpretivism), and so, this work adopts a pragmatic approach under a mixed-method case study design (refer to Section 3.3 and Section 3.4). Under the case study method, relevant software artefacts are studied using a multi-phase approach in alignment with the general principles of data mining best practice, social network analysis, linguistic analysis and directed content analysis (refer to Section 3.4). Through the use of the IBM Rational Jazz repository, data mining practices were used for data extraction and preparation. The other aforementioned techniques (social network analysis, linguistic analysis and directed content analysis) are then used to study the collaboration patterns of successful globally distributed agile teams and how and why core developers contribute to globally distributed agile team dynamics based on these artefacts.

In alignment with the theoretical stance of similar empirical work, these outcomes are aimed at providing initial conjectures that may lead to explanation theories. In Section 3.5 it is noted that this form of theory conforms to Gregor's Type 2 classification. This theoretical approach and the work's overall methodological framework are further illustrated in Figure 18.

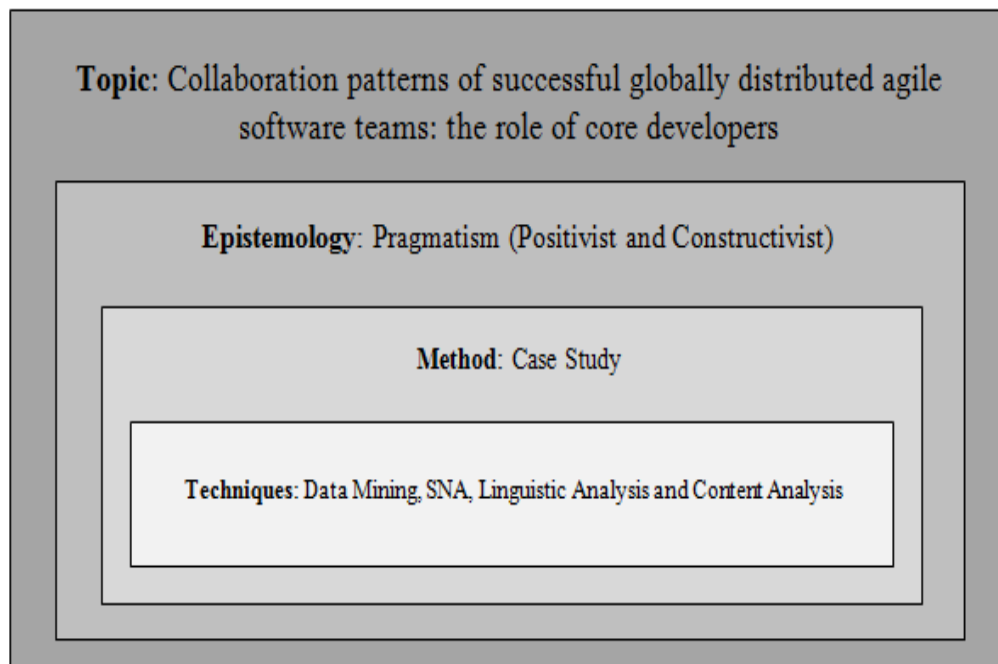


Figure 18. The methodological framework of this PhD

Chapter 4. Results

This chapter reports the results of the research, and comprises four sections. Firstly, Section 4.1 presents the preliminary social network analysis results that form the confirmatory analyses, and is aimed largely at providing the foundation for the other main sections (Section 4.2 and Section 4.3) that follow. These latter two sections comprise the exploratory analyses in alignment with the discussions in Section 3.4. In incrementing the results in Section 4.1, Section 4.2 is next outlined; this section provides a static (single project snapshot) view of the linguistic and directed CA results that were revealed in this work, and also forms the basis for the subsequent section - Section 4.3. This section (Section 4.3) considers the linguistic and directed CA results from a temporal perspective. Finally, Section 4.4 provides a summary of the results that are presented throughout this chapter.

4.1 Phase 1 – Social Network Analysis

This section considers the preliminary results aimed at answering research questions RQ1 – RQ5. These preliminary results relate to the exploration of collaboration patterns of successful globally distributed agile teams. Firstly, Section 4.1.1 explores the teams' contributions of messages over their project and addresses RQ1: Do communication patterns change as the software project progresses? Section 4.1.2 then provides the social network analysis results for the ten teams selected. These results include those related to the SNA metrics outlined in Chapter 3 and the visualisations of teams' sociograms, and are aimed at answering RQ2: Is there equity in practitioners' contributions to their project? RQ3: Are active communicators more important to their teams' collaboration?, RQ4: How are active communicators involved in task performance ? and RQ5: Are practitioners' formal role assignments related to their involvement in project interactions and task performance?, are then answered by the results that are presented in Section 4.1.3.

4.1.1 Project Communication Patterns (RQ1)

This section presents the results obtained from the preliminary analysis of Jazz teams' interactions, which is aimed at answering RQ1. Given the purposive sampling approach adopted, minimal detailed comparisons of communications *across* the teams are made. Rather, communications *within* teams are examined in this section. First, Jazz teams' communications are explored to understand how these teams typically interacted as

their project progressed. This enquiry is aimed at gaining familiarity with the Jazz project interaction environment, and is used to set the tone for the deeper and specific explorations conducted subsequently in relation to active developers (see the approach used by Shihab et al. (2010) in their examination of the communication style of practitioners in GNOME, for an example of a similar analysis). Adoption of such an approach to explore within-project interactions as the teams progress is also useful for observing and understanding any temporal changes, as against simply observing a static view of project communication (Hinds & McGrath, 2006). Yu et al. (2011, p. 223) employed a similar approach to study GTK+ project interactions. Similar analyses of interaction evolution were also conducted by others (Cataldo et al., 2006; Datta et al., 2011; Shihab et al., 2009).

As noted in Section 3.4.3.1, each set of team artefacts are divided into four equal parts (start, early-mid, late-mid, and end) to account for differences in the number of iterations and duration in each project area (note that the lowest number of iterations for the individual teams (P1 – P10) was two (for P3) and the highest was 17 (for P5) – see Table 2). In Table 2, the total number of messages communicated by the selected teams is shown to be 5563. Overall, over the four project phases, 1549 messages (the highest number) were contributed in the start phase, 1333 messages were communicated in the early-mid phase, 1263 messages (the lowest number) were communicated in the late-mid phase and 1418 were contributed in the end phase. (Refer to Table 8 for descriptive statistics concerning practitioners' communication over the course of the Jazz project.)

Table 8. Descriptive statistics for combined teams' (P1 – P10) communication

Phase	Messages (N)	Mean Messages/WI	Median Messages/ WI	SD	SK	KS	Std. Error of SK	Std. Error of KS
start	1549	6.1	4.0	6.0	2.6	9.6	0.2	0.3
early-mid	1333	5.2	3.0	5.9	3.0	11.8	0.2	0.3
late-mid	1263	4.9	3.0	5.4	2.8	9.9	0.2	0.3
end	1418	5.5	3.0	5.1	1.8	3.2	0.2	0.3
Mean	1391	5.4	3.3	5.6	2.6	8.6	0.2	0.3

Notes: SD = Standard Deviation, SK = Skewness, KS = Kurtosis

In order to verify whether there were any significant differences in the way teams interacted over the four project phases a series of statistical tests was conducted. Firstly, checks for normality of the data distributions were conducted using Kolmogorov-Smirnov tests (given that there were > 1200 messages in each phase) (see Brooks, Clarke, & McGale (1994) for discussions on the formal application of normality tests).

These tests confirmed that the distributions of messages communicated around software tasks violated the normality assumption for all four project phases. The standardised skewness coefficient (i.e., the skewness value divided by its standard error) and standardised kurtosis coefficient (i.e., the kurtosis value divided by its standard error) were also outside the boundaries of normally distributed data (i.e., -3 to +3) (Onwuegbuzie & Danel, 2002), see Table 8 for details. Thus, the nonparametric Kruskal-Wallis test was used to check for differences in communication across the four project phases.

The Kruskal-Wallis test revealed that there was a statistically significant difference in the numbers of messages communicated over the four project phases $X^2 = 12.596$, $p < 0.01$. The effect size associated with this difference, as measured by Cramer's V, was 0.227. Using Cohen (1988) criteria, this measure is indicative of a small effect size. A series of four Mann-Whitney pairwise follow-up tests at the Bonferroni adjusted level (Vogt, 2005) (i.e., 0.05 divided by 4 analyses) of 0.0125 indicated that Jazz teams communicated significantly more at the start of their project than in the early-mid ($p < 0.0125$) and late-mid ($p < 0.0125$) phases. Comparisons between the early-mid and end phases, and late-mid and end phases, did not produce statistically significant results ($p > 0.0125$ and $p > 0.0125$ respectively).

A detailed view of the teams' interactions over their project is presented through an examination of the mean messages communicated per WI by practitioners for each team (P1 –P10) in Figure 19 (see median messages communicated in Appendix I where a similar pattern of results is obtained). Figure 19 shows that for seven teams (P2, P3, P4, P5, P8, P9, and P10), there were elevated levels of messages in the first phase, and messages also increased towards project completion for most teams (except for teams P1 and P4). These results are in line with those reflecting the overall project measures. With the exception of team P5 (where communication increased in the third phase), all teams also recorded a reduction in communication over the second and third project phases – a pattern also evident in the overall results.

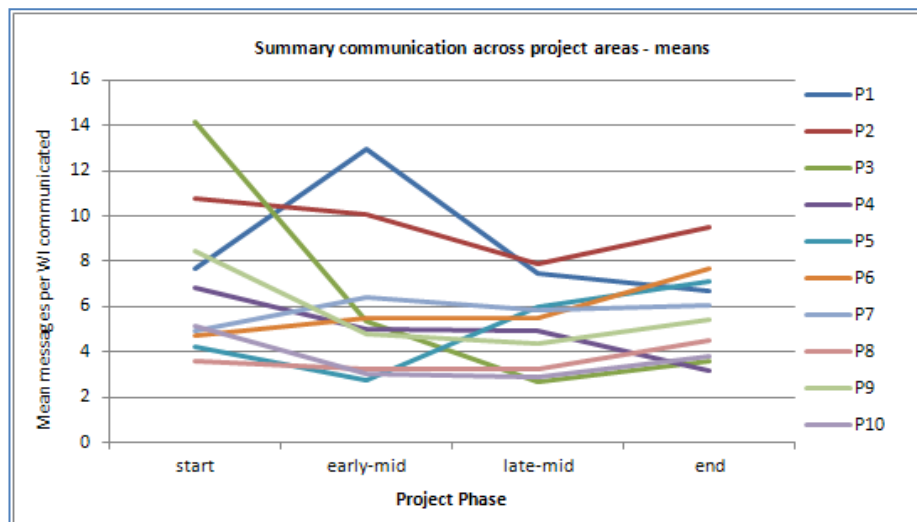


Figure 19. Mean messages per WI communicated over project phases (teams P1– P10)

The above results reveal that practitioners communicated the most in the start and end phases of their project. This finding is divergent to those revealed in previous work (Datta et al., 2011), which found developers' communications increased as the project progressed, before stabilising towards project completion. This divergence suggests that different communication strategies may be adopted by different teams given the specific method that is used for developing software. For example, approaches that are aimed at capturing most of the software requirements at project initiation would likely demand heavy communication in the early project phases, while those that encourage 'continuous' requirements solicitation as projects progress may see teams communicating more consistently throughout the project or may experience higher levels of communication during the middle project phases. While these results provide a view into ten Jazz teams' communications, aggregating and then analysing the teams' messages as has been done above does not reveal the *internal* interaction patterns of these teams. SNA techniques may be used for this activity (De Laat et al., 2007; Wasserman & Faust, 1997). Such analyses are conducted in the next section, where Jazz teams' communications are explored using both sociograms and SNA metrics (refer to Section 3.4.3 for discussions around these techniques).

4.1.2 Equity in Practitioners' Communication (RQ2)

The results in this section are aimed at answering RQ2. As noted in Chapter 3, sociograms (or task based social networks – refer to Wolf, Schroter, Damian, Panjer, & Nguyen (2009) for a discussion on the utility of this method for studying team interactions around software tasks) were created based on the messages conveyed and tasks undertaken by each of the ten teams (P1 – P10). Qualitative visual examination of

the teams' network graphs reveal that, for all ten project areas, just a few practitioners dominated communication (see Figure 12 for example; Appendix II provides the sociograms of all ten Jazz teams considered here). Additionally, when examining the SNA in-degree measures for the project members of the ten teams (see Figure 20 for in-degree measures for a sample Jazz team), it is noticed that these measures were highly skewed for all ten teams. Accordingly, formal statistical testing is conducted. First, the distribution for each team's in-degree measures is examined for adherence to the normality assumption using the Shapiro-Wilks test (given that the samples (P1 – P10) all comprised fewer than 100 contributors), and the standardised skewness and kurtosis coefficients is also considered (see Section 4.1.1 for details). These tests corroborate that the distributions violated the normality assumption for all ten teams; and thus, confirm that in-degree measures were skewed for all the teams (see Table 9 for descriptive statistics concerning teams' in-degree measures). Given this violation of the normality assumption, the Kruskal-Wallis test is used to test for differences in in-degree measures across the ten Jazz teams. The result of this test reveal that Jazz teams' in-degree measures were indeed relatively homogenous (i.e., few practitioners dominated interactions for all ten Jazz teams), $X^2 = 13.182$, $p > 0.05$.

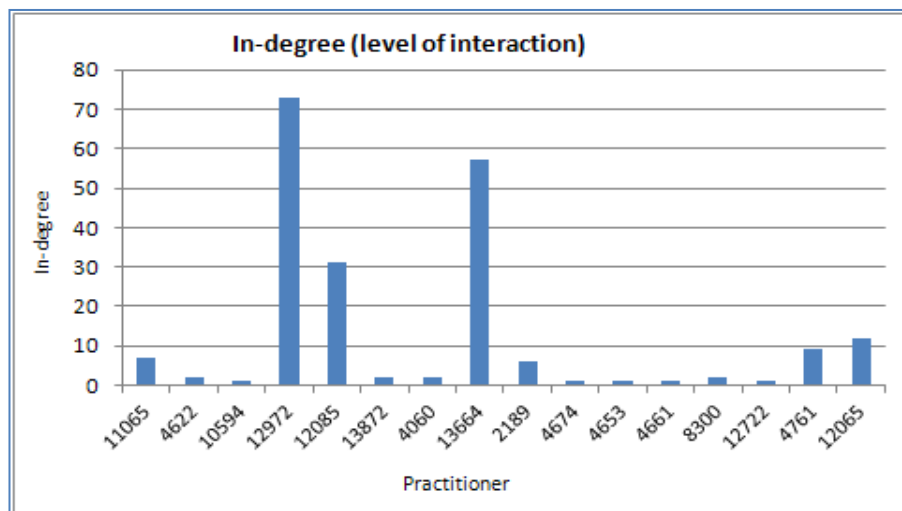


Figure 20. In-degree measures for a sample Jazz team

Miller, Daly, Wood, Roper, & Brooks (1997) noted that homogeneous samples such as these tend to yield statistically powerful and reliable results. This pattern of significant skewness in terms of team members' interactions was also observed by Shihab et al. (2010) and Yu et al. (2011), and in part set the tone for the further work that is performed in this study. Particularly, this leads to questions being raised such as why such a pattern might exist, and what are the implications of this pattern for software engineering team dynamics? These observations point to a need to scrutinise and

explore these highly active individuals further, to provide insights into their behaviours and inferences for software development. Thus, in the current preliminary SNA investigation, dominant contributors are examined further, and these results are presented in Section 4.1.3. The following results pertain to the ten teams' networks (P1 – P10); here an initial examination of teams' involvement in communication, and the evolution of the ten teams' sociograms over time, are provided through the SNA lens.

Table 9. Descriptive statistics for teams' in-degree measures (P1 – P10)

Team ID	In-degree	Mean	Median	SD	SK	KS	Std. Error of SK	Std. Error of KS
P1	141	4.3	2.0	8.8	4.1	17.6	0.4	0.8
P2	249	5.3	2.0	13.0	5.2	29.4	0.3	0.7
P3	68	2.3	2.0	2.7	3.9	17.1	0.4	0.8
P4	438	11.2	2.0	20.6	2.5	5.4	0.4	0.7
P5	296	6.2	3.0	10.4	3.4	12.2	0.3	0.7
P6	230	9.2	2.0	19.5	3.2	9.7	0.5	0.9
P7	208	13.0	2.0	21.9	2.1	3.7	0.6	1.1
P8	374	4.2	1.5	8.1	4.7	26.1	0.3	0.5
P9	114	6.0	1.0	9.2	1.7	1.2	0.5	1.0
P10	365	7.6	3.0	19.0	5.6	34.9	0.3	0.7
Mean	248.3	6.9	2.1	13.3	3.6	15.7	0.4	0.7

Notes: SD = Standard Deviation, SK = Skewness, KS = Kurtosis

As observed in Section 4.1.1, there were higher levels of communication in the start and end phases of the Jazz project. Qualitatively examining the ten teams' sociograms reveals a similar pattern (see Figure 21 for one team's set of sociograms). Additionally, the network structures, in terms of individual contributors' interactions, were generally stable over project duration, with communication structures established in the early and middle stages of the project being preserved throughout the project. The underlying trend that is observed here is that those who interacted little at the beginning of the project remained relatively quiet throughout the project, whereas high communicating members did so throughout, a phenomenon also observed by Shihab et al. (2010). This is demonstrated in Figure 21 where snapshots of one of the Jazz teams' social networks over the four project phases are presented. These graphs show that in the start phase of the project (graph One) practitioners 6262 and 13722 occupied hubs for team communication, and communicated densely on the tasks in which they were involved. This pattern continued in the early-mid and late-mid project phases (graphs Two and Three, respectively), where it is noted that only a few other individuals contributed significant numbers of messages. The fourth graph demonstrates that this behaviour is also maintained in the end phase, and practitioners 6262 and 13722 continued their

dominant communication patterns throughout the project (see Appendix II for sociograms of all ten Jazz teams considered over the four project phases).

As noted above, this work has employed a purposive sampling approach, and so formal comparisons of the measures *across* the teams selected are not performed. That said, the fact that all the artefacts were selected from a single repository, and that this work employs multiple analysis approaches (both top-down and contextual) in analysing a large number of artefacts, make some level of comparison valid (Creswell, 1998; Kuzel, 1992). Additionally, Romney et al. (1986) showed that samples comprising as few as four individuals can render highly accurate and generalisable information if the individuals are very competent in the domain under investigation, as is contended here regarding the Jazz teams (noted in Chapter 3). Therefore specific comparisons across the teams are now performed.

It is evident in Table 2 that team P8 (centred on project management-based tasks) was most heavily populated with contributors. For this project area 90 members contributed 612 messages to the network. Although this team shares similar characteristics with teams P4 and P10 for task count (P4 = 214 tasks, P8 = 210 tasks and P10 = 207 tasks, respectively) and the number of messages contributed overall (P4 = 883 messages and P10 = 640 messages; see Table 2 for further details), the number of contributors on project area P8 (90 members) is double those in project areas P4 and P10 (these have 39 members and 48 members, respectively). Given this volume of contributors, it could be expected that communication within this team would be the least dense of all the teams P1 – P10. While the result confirms that this was indeed the case for P8 (with a density measure of 0.02), Figure 22 (a) shows that all of the project areas that are observed lacked high levels of density – the highest density of all the teams is observed for P7, being just over 0.14). Note in Section 3.4.3 that density varies between 0 and 1, where a task that attract communication from the entire team would have a density of 1, while one with no communication would have a density of 0. Thus, density of less than 0.3 may generally be considered low. The overall Jazz project measures for density are affected by the generally low level of messages contributed by team members, the exception being the more active contributors – the mean measure is 0.07 (median = 0.07 and SD = 0.04). Many of the contributors communicated on a few tasks only (as is illustrated in Figure 12 and Table 9). Implementation- or functionality-centric teams (project areas P7 and P9) saw the highest levels of distribution of contributions across software tasks; with density values of 0.14 and 0.12, respectively (see Figure 22 (a)).

The teams' overall measures for density are in contrast to those for closeness centrality where measures for all teams were very close (see Figure 22 (c)); the mean closeness measure for the project areas (P1 – P10) is 0.01 (median = 0.01 and SD = 0.02). The least close team (with a closeness value of 0.06) was project area P3 (documentation-based), which also took the shortest time to tasks completion (59 days) and had the lowest numbers of tasks (30 WIs) and communications (158 messages) (see Figure 22 (c) and Table 2). With the exception of team P3, all the project areas had a closeness centrality measure of less than 0.02, denoting very close networks (see Hanneman & Riddle (2005) and Wasserman & Faust (1997) for further details on closeness centrality). Such closeness measures denote that practitioners remained accessible, whether directly or through their connections, across all the project teams. In project areas P4, P8 and P10, practitioners' contributions were highest (with in-degrees 438, 374 and 365, respectively) (see Figure 22 (d)), although, on average, these teams did not observe very high numbers of contributions across software tasks (mean in-degree (P1 – P10) = 248.3, median = 248.3 and SD = 120.9). The representation in Figure 22 (b) of measures for mean unique edges confirms these measures. In fact, in examining Figure 22 (b) it is evident that the average number of unique contributions across software tasks in the individual teams was quite stable (mean unique edges (P1 – P10) = 2.2, median = 2.2 and SD = 0.3).

The results just reported all serve to indicate that only a few individuals communicated per team, across all ten project areas. These specific individuals maintain this distinct presence in their teams' communication networks throughout the Jazz project. This finding suggests that the sample of project areas selected was homogenous (Miller et al., 1997). In fact, although the teams were dedicated to addressing different forms of software task, there was general consistency in the SNAs' density, unique edges, closeness and in-degree measures. Overall, the analyses conducted in this section (and their associated visualisations) reveal that there is inequity in communication for all ten Jazz teams. However, these results do not reveal the *scale* of the inequality in practitioners' communications (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Cataldo et al., 2006; Shihab et al., 2009), nor do they answer the questions “Why are only a few developers communicating in all ten teams?”, “What are the roles of these active communicators?” and “How are these active communicators involved in task performance?”. Previous research has found that centralized patterns involving core group members are a positive sign for team performance (Ahuja et al., 2003; Bavelas,

1950; Guetzkow & Simon, 1955). Central individuals are also generally seen as projects' leaders, whether or not they are the formal leaders (Leavitt, 1951), and groups with central coordinators experience higher levels of group organization and task performance (in terms of speed of completion). Accordingly, the communication patterns of active contributors are further scrutinised in the next section, as a first step to understanding the reasons for these practitioners' distinct presence.

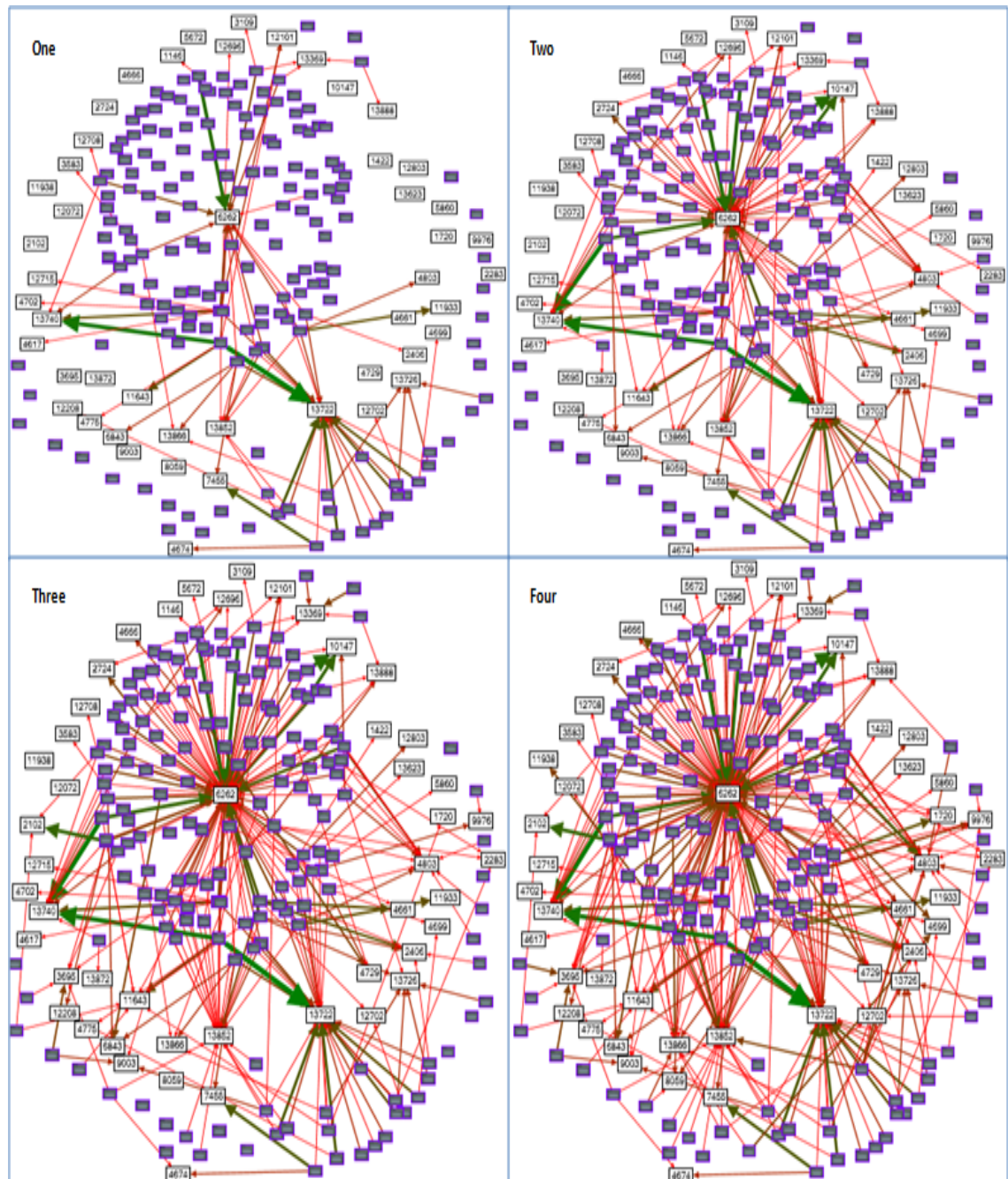


Figure 21. Jazz sample team network (phases one (start) to four (end))

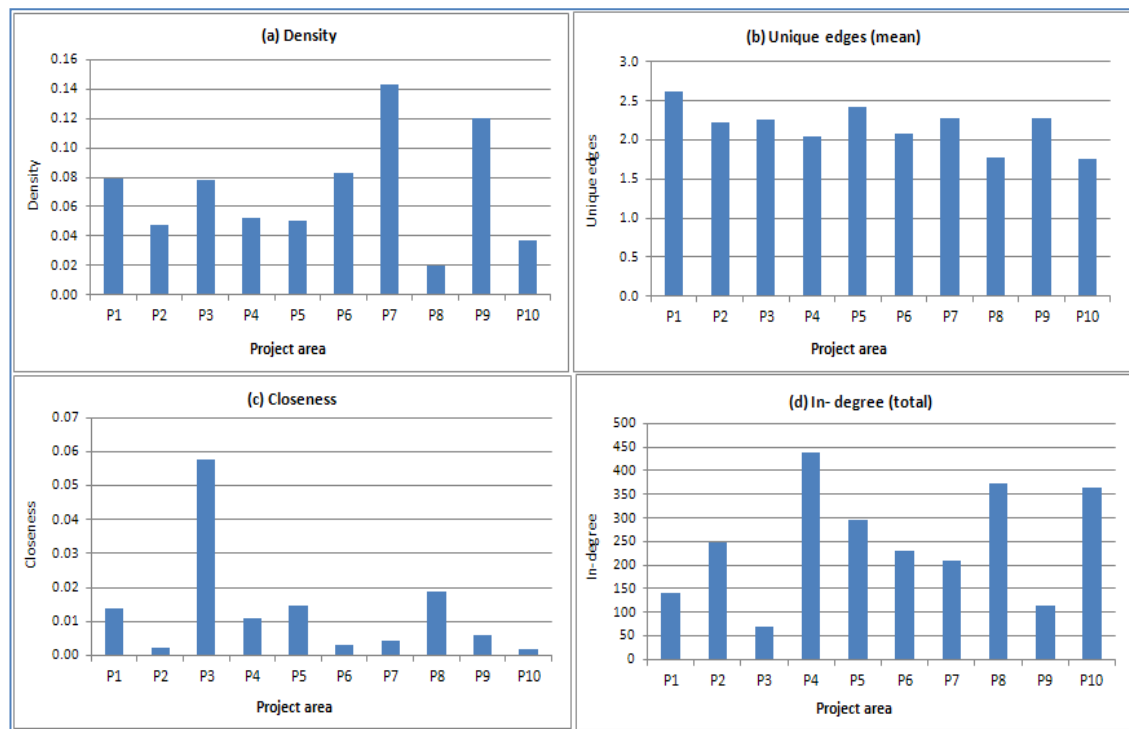


Figure 22. Summary of social network measures for the ten project areas (P1 – P10)

4.1.3 Importance, Task Performance and Formal Roles (RQ3, RQ4 and RQ5)

As highlighted previously, in all ten project areas (P1 – P10) just a few individuals dominated the interaction networks (see Appendix I (b)). As intimated towards the end of Section 4.1.2, individuals involved in highly dense communication network segments, such as those observed here, have previously been shown to occupy the centre of group coordination and collective action (Reagans & Zuckerman, 2001), and are seen as most important to their teams' knowledge sharing processes (Leavitt, 1951; Zhong et al., 2012) (see Chapter 2). While previous research has observed this pattern (Cataldo et al., 2006; Shihab et al., 2009), questions related to *how* and *why* core group members become 'knowledge hubs', the reasons for these members' extraordinary presence, and understanding the actual roles (both formal and informal) that core developers occupy in their teams, have not been answered. Such answers could provide explanations for the nature and peculiarities of distributed agile group dynamics. Knowledge of the way the most active agile practitioners contribute their social and intellectual capital could help project leaders to identify exceptional software practitioners early, and inform the process of assembling high performing and cohesive teams. Such findings could also inform the use of specific organizational arrangements and team configurations in support of high performers. Furthermore, the output of these explorations may lead to new requirements for collaboration and process support tools.

In addressing these research gaps and opportunities, the artefacts of core developers are deliberately targeted for exploration in order to answer RQ3, RQ4 and RQ5. As outlined in Chapter 3, core developers were selected using an initial baseline density measure of ≥ 0.33 (i.e., they communicated on a third or more of their teams' project tasks) (Crowston et al., 2006). Table 10 shows that only fourteen contributors across the ten project areas met this initial density-based selection criterion for core developers (shown as bold font contributor numbers – notice that none of the members from P8 were selected initially). Thus, the top two contributors to each team were instead selected (an approach also employed by Rigby & Hassan (2007)), which increased the total number of core developers by six (the non-bold font contributor numbers), bringing the core developers cluster to 20. Note also from Table 10 that a few core developers were dominant across multiple project areas (e.g., see contributors 4661 and 2419 for P1 and P2 in Table 10). Thus, in total there were 15 distinct core developers.

Table 10. Social network measures for core developers and their team scores (P1 – P10)

Team ID/ Project area	Contributor	Core In-degree		Density		Closeness	
		In-degree	(% of team measure)	core	team	core	team
P1	4661	46	32.6	0.85	0.08	0.01	0.01
	2419	26	18.4	0.48		0.01	
P2	4661	83	33.3	0.74	0.05	0.00	0.00
	2419	33	13.3	0.29		0.00	
P3	13722	15	22.1	0.50	0.08	0.01	0.06
	4674	7	10.3	0.23		0.01	
P4	13740	85	19.4	0.40	0.05	0.00	0.01
	11643	70	16.0	0.33		0.00	
P5	4749	55	18.6	0.45	0.05	0.00	0.01
	4674	39	13.2	0.32		0.00	
P6	12065	82	35.7	0.74	0.08	0.01	0.00
	13664	61	26.5	0.55		0.00	
P7	12972	73	35.1	0.80	0.14	0.01	0.00
	13664	57	27.4	0.63		0.01	
P8	12702	59	15.8	0.28	0.02	0.00	0.02
	2102	33	8.8	0.16		0.00	
P9	6572	29	25.4	0.58	0.12	0.01	0.01
	12889	22	19.3	0.44		0.01	
P10	6262	127	34.8	0.61	0.04	0.00	0.00
	13722	36	9.9	0.17		0.00	
Mean	-	51.9	21.8	0.48	0.07	0.00	0.01

Table 10 shows that these core developers were indeed actively involved in their teams' communication. In terms of in-degree, core developers contributed over 62% of the teams' measures for P6 and P7, and core developers on P1, P2, P9 and P10 contributed

a combined 51.1%, 46.6%, 44.7% and 44.7% of their teams' measures, respectively. Overall, the core developers had a mean in-degree score of 51.9 (refer to Table 10). This value is substantial when considering the mean in-degree score for the ten project areas (P1 – P10) was 248.30 (see Table 9). In fact, this number represents 21.79% of the overall measure for all project areas (refer to Table 10). The density figures in Table 10 show a similar pattern. Here it is revealed that contributors 4661 (of P1), 12972 (of P7) and 12065 (of P6) had density measures of 0.85, 0.80 and 0.74 respectively, and the overall mean density measure for core developers was 0.48, compared to the mean project areas' (P1 – P10) density measure of 0.07 (i.e., core developers communicated on 48% of the tasks compared to their overall teams' score of 7%). These overall project areas' measures are compared for statistically significant differences. When the density scores are checked for normality it is noted that there is no violation of the normality assumption (Onwuegbuzie & Danlel, 2002). A Levene's test for equality of variance reveal unequal variances for the two groups (core developers and their team) ($p = 0.001$). Thus, the parametric independent sample t -test is conducted to test the mean density measures for significant differences. This results show statistically significant difference between the density of core members and those of their overall project areas', ($t(9.95) = 7.85, P = < 0.001$). This difference represent a large effect size (Cohen's d) of 0.928 (Cohen, 1988). This result show core developers communicated on nearly seven times as many software tasks as their teammates (see Figure 23 (a–d) for additional visualisations).

In considering the closeness column in Table 10, a different pattern is observed. Noticeably, the network measures for closeness for core developers are not much lower than those for the teams' networks (with an overall mean score of 0.00 for core developers and 0.01 for the teams). In fact, Table 10 shows that for project area P7 the closeness measure for the overall team network is lower than those for the central players (0.00 versus 0.01 for each core developer). The closeness scores are checked for statistically significant differences. Firstly, an examination of the standardized skewness and kurtosis coefficients for the closeness measures reveal serious departures from normality (Onwuegbuzie & Danlel, 2002). Thus, a non-parametric Mann-Whitney U test is used to compare the scores of the core developers and those of their teams (refer to Table 10). This test does not reveal any statistically significant difference in the closeness scores for core developers and their teams, ($U = 40, P = 0.404$). These findings are supported by an examination of Figure 23 (e–f) which show that even with

the removal of the core developers from the teams' networks, most contributors still remain reachable (whether directly or via others in the networks). These findings denote that, overall, Jazz members were all very accessible irrespective of their levels of active contribution.

Communication frequency and volume may be linked to individuals' assigned roles (Shihab et al., 2010). Given this, roles more inclined to coordination or liaison, such as project manager and project administrator, might be expected to be more heavily involved in project communication. Members occupying these and similar roles may not necessarily be actual core developers on software tasks (Cataldo & Herbsleb, 2008) – in terms of their efforts in problem solving. Accordingly, core developers' communication behaviour is checked against their formal roles and their actual involvement with software tasks (task performance). Firstly, core developers' actual role information is extracted from the repository. This data show that eight out of the 15 distinct cluster members were programmers, along with five team leaders and two project managers. This evidence provides some level of support for the preliminary assessment that the members clustered in the core developers group were high contributors (in terms of knowledge contribution) on software tasks. As noted in Section 3.4.2, in Jazz a person occupying the formal "Programmer" (contributor) role is defined as a contributor to the architecture and code of a component, the "Team Leader" (component lead) is responsible for planning and architectural integrity of the component and the "Project Manager" (PMC) is a member of the project management committee overseeing the IBM Rational Jazz project.

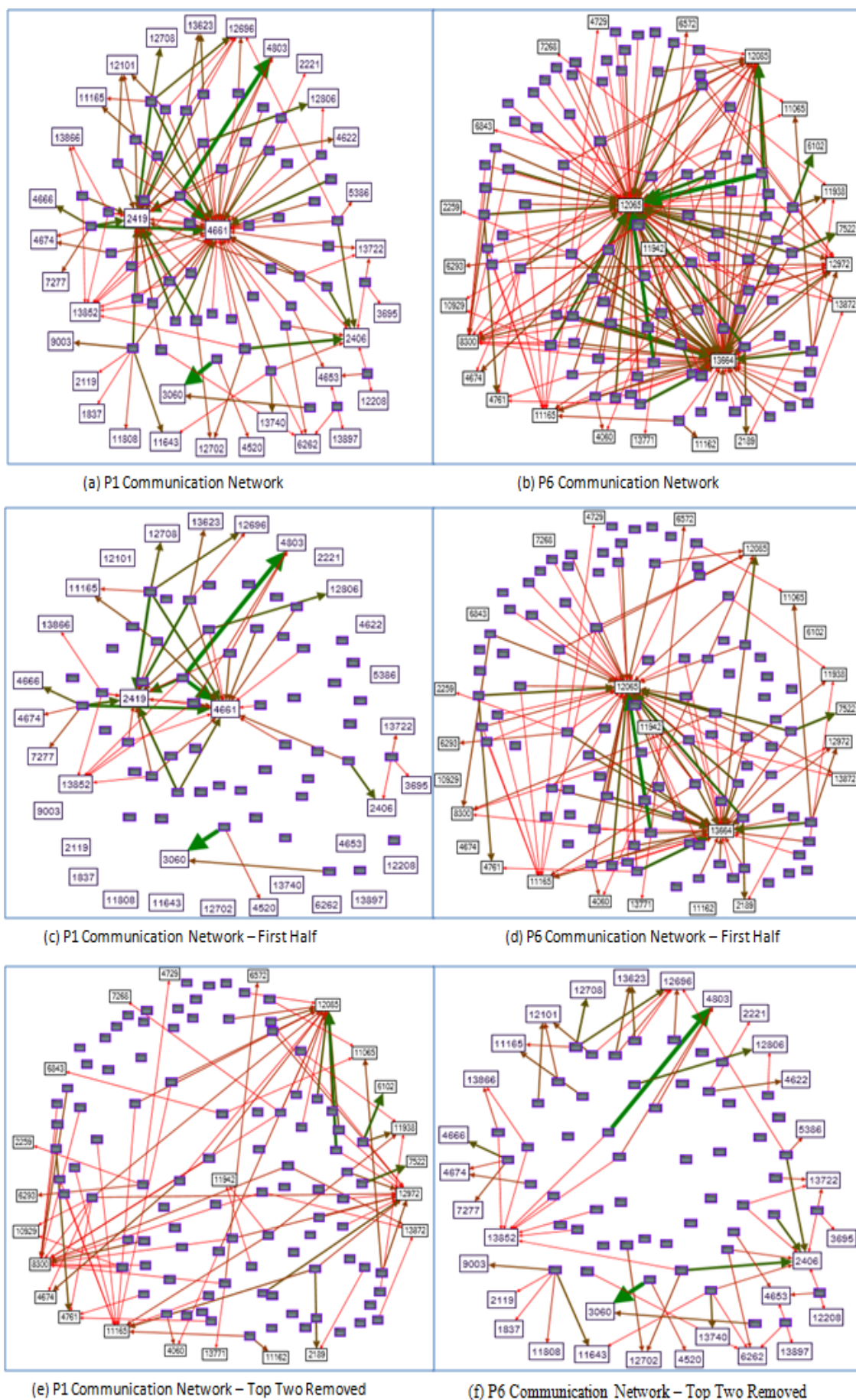


Figure 23. Sample network graphs (sociograms)

Chapter 3 described the approach that is used for studying these contributors' task performance (Cataldo & Herbsleb, 2008; Shihab et al., 2010), where it was noted that the mined change logs were summarised (Cataldo & Herbsleb, 2008). Having confirmed the formally assigned roles of the core developers, the third step in this phase of the analyses is to examine core developers' involvement in software development tasks. Table 11 shows that, on average, more than 41% of all software tasks were initiated by the 15 core developers. These practitioners also made more than 69% of the changes to these tasks and resolved nearly 75% of all software tasks that were undertaken by their teams. In fact, core developers created as many as 69% of all software tasks in P6 and made 94% of changes in P9 (refer to Table 11). These scores were further exceeded in project area P9, where core developers resolved 98% of their team's tasks. These figures are in contrast to what would be a 'typical' contribution if WIs were distributed evenly across all contributors to a project area – taking this latter approach, team members would typically have contributed to between 1.1% (for P8) and 6.3% (for P7) of their teams' WIs changes (see Table 2). These task change results provide support for the approach that was employed to select the core developers; the members selected in the core developers cluster were truly the most active members on their projects (Cataldo & Herbsleb, 2008). These task change results are also triangulated with contextual analyses of these members' (and their teammates') messages.

Table 11. Activities performed by core developers

Team ID	% Created	% Modifications	% Resolved
P1	44.4	66.7	79.6
P2	49.1	58.0	67.0
P3	26.7	66.7	20.0
P4	36.0	49.1	60.7
P5	16.4	62.3	73.0
P6	65.8	78.4	97.3
P7	44.0	63.7	91.2
P8	28.6	73.3	64.3
P9	60.0	94.0	98.0
P10	39.6	85.0	93.7
Mean	41.1	69.7	74.5

The findings just reported endorse those discovered previously, that only a small number of team members tend to contribute to a software project's knowledge base (Shihab et al., 2010), and that software developers' communication and coordination activities are directly related to their involvement in software tasks (Bird et al., 2006a).

These results, and particularly those related to core developers' in-degree and density scores, show that core members communicated on significantly more tasks ($p < 0.001$) than the other members of their teams. The Cohen's d effect size associated with the difference in density scores between core developers and the other practitioners was large (Cohen, 1988), suggesting that the difference noted is of practical importance (Kampenes, Dybå, Hannay, & Sjøberg, 2007).

This pattern was also previously revealed by others studying software teams' communication (Bird et al., 2006a; Shihab et al., 2010; Yu et al., 2011). However, researchers have tended to stop there, and have not investigated further as to why this phenomenon exists. In particular, evidence around why core developers demonstrate such a distinct presence and how these members evolve into their central roles have not been provided. Such evidence could be integral for informing team composition strategies, and may also inform software tool features in support for team governance. Accordingly, linguistic analysis of Jazz practitioners' messages is conducted as a first step towards answering these questions; these results are presented in the next section.

4.2 Phase 2 – Linguistic Analysis and Directed CA (Static Analyses)

Results in this section are aimed at establishing whether core developers' attitudes differ from those of 'regular' (i.e., non-core) team members, and examining the roles enacted by core developers. These enquiries provide understandings for the reasons for core developers' very high levels of communication and task performance. Additionally, the results around core developers' expression of attitudes and contribution of knowledge are also intended to explain the specific nature of the way these members contribute to their teams' dynamics. Firstly, Section 4.2.1 reports the exploration of the attitudes of core developers and a comparison of the behaviours of these practitioners against those of their less active counterparts. This aspect of the results is aimed at answering RQ6 (Do core developers' behaviours and attitudes differ from those of other software practitioners?). Section 4.2.2 provides a similar comparative analysis through a directed CA (qualitative or contextual) lens. The directed CA results are aimed at answering RQ7 (What are the core developers' enacted roles in their teams, and how these roles are occupied?), towards establishing what roles core developers enact in their teams. Both forms of analyses, linguistic and directed CA, consider the project areas in the form of a single snapshot.

4.2.1 Attitudes (RQ6)

In Section 3.4.3.1 the procedure used for selecting the core developers was introduced (Crowston et al., 2006; Rigby & Hassan, 2007). Summaries of the results for core developers' communication (refer to Figure 21 and Figure 23 for illustrations) and their involvement in task performance are provided in Table 10 and Table 11, respectively. These results show that core developers communicated the most and were also integral to their teams' actual software development portfolio. Thus, active communicators were not merely team coordinators. It was also discovered that core developers were not restricted by their formal roles, as quite often programmers leading their teams' communication worked under formal project leaders (refer to Section 4.1.3). These results are extended in this section, and the attitudes of core developers are examined using a psycholinguistic approach to answer RQ6. The attitudes that are commonly expressed by core developers' are also compared to those expressed by their counterparts. This is achieved using an analysis of the content of the messages contributed by core developers and the other practitioners, using the predefined linguistic dimensions in Table 5. These dimensions were used previously to study behaviours expressed in textual communication (Mairesse & Walker, 2006; Pennebaker et al., 2007; Pennebaker & King, 1999; Rigby & Hassan, 2007).

All communications contributed by the two groups of developers (core and others) are aggregated. Those 15 practitioners classified as core developers from the total of 146 distinct practitioners across the ten project areas contributed 2565 messages out of the total 5563 messages shown in Table 2. Given the sample sizes, with both groups contributing > 2500 messages, the form of the data distributions is first evaluated by analysing the messages in the two groups along the 13 linguistic dimensions using the Kolmogorov-Smirnov test of normality. The data for both sets of messages show violations of the normality assumption. The standardised skewness and kurtosis coefficients are also outside the boundaries of normally distributed data (i.e., -3 to +3) (Onwuegbuzie & Danel, 2002) – see Table 12 for a summary of the descriptive statistics for the linguistic scores of core developers and others. Accordingly, paired (core and others) comparisons are conducted for the individual linguistic dimensions to check for significant differences using the non-parametric Mann-Whitney U test. These results are presented in Table 13.

Table 12. Descriptive statistics for linguistic scores for core developers and others

Linguistic Category	Abbrev.	Mean		SD		SK		KS	
		Core	Others	Core	Others	Core	Others	Core	Others
Pronouns	I	7.6	8.3	15.0	15.1	3.3	3.1	13.9	13.2
	we	2.5	2.7	6.2	6.0	3.7	2.9	17.2	9.8
	you	2.8	2.2	6.7	5.7	3.2	3.9	12.4	19.4
Cognitive	insight	5.7	6.4	11.4	11.5	3.8	3.3	22.0	17.5
	discrep	6.0	5.9	10.9	10.2	3.6	3.1	21.5	17.0
	tentat	5.2	6.0	10.7	10.8	4.2	3.3	27.0	18.3
	certain	2.4	2.8	7.7	8.9	6.9	6.6	68.2	58.2
Work and Achievement	work	12.5	10.3	18.6	15.8	1.7	1.9	2.6	4.0
	achieve	11.4	9.8	17.1	15.1	1.7	2.0	2.1	4.2
Leisure, social and positive	leisure	2.5	2.9	7.7	8.3	5.5	5.8	43.8	47.3
	social	12.9	13.1	16.5	16.4	1.7	1.9	4.7	5.8
	posemo	19.9	21.8	32.3	34.0	1.8	1.6	1.7	1.1
Negative	negemo	3.4	3.3	11.0	10.1	5.9	5.6	43.6	40.2

Notes: SD = Standard Deviation, SK = Skewness, KS = Kurtosis

Table 13. Results for linguistics analysis

Linguistic Category	Abbrev.	Core (Mean Rank)	Others (Mean Rank)	Mann-Whitney Test (<i>p</i> -value)
Pronouns	I	2711.5	2853.3	0.000
	we	2752.7	2818.2	0.039
	you	2829.9	2752.3	0.012
Cognitive	insight	2716.7	2848.9	0.000
	discrep	2779.1	2795.6	0.663
	tentat	2706.1	2857.9	0.000
	certain	2742.2	2827.1	0.007
Work and Achievement	work	2841.8	2742.1	0.013
	achieve	2828.2	2753.7	0.063
Leisure, social and positive	leisure	2738.6	2830.1	0.003
	social	2772.8	2801.0	0.490
	posemo	2748.1	2822.1	0.073
Negative	negemo	2773.9	2800.0	0.410

Table 13 shows that core developers were less self-focused, in that they used less individualistic language (e.g., I, me, my) than the other contributors, and they tended to delegate more (e.g., you, your, you're). The Mann-Whitney U test comparing these language dimensions for the two groups confirmed that these differences are statistically significant ($p < 0.001$ and $p < 0.05$, respectively). The Cohen's *d* effect sizes associated with these differences are 0.050 and 0.034 respectively. These findings represent small effect sizes (Cohen, 1988). The other team members used significantly more individualistic language, and this group also used significantly more collective language (e.g., we, our, us) ($p < 0.05$). This finding for collective language use has an effect size of 0.030, also small.

The other team members also used significantly more insightful (e.g., think, believe, consider) ($p < 0.001$), tentative (e.g., maybe, perhaps, apparently) ($p < 0.001$) and certainty (e.g., definitely, extremely, always) ($p < 0.01$) type utterances. The Cohen's d effect sizes associated with the differences for these three cognitive dimensions (insight, tentative and certain in Table 13) are all small, being 0.050, 0.050 and 0.040, respectively. This pattern is the opposite for work (e.g., feedback, goal, delegate) and achievement (e.g., accomplish, attain, resolve) related language use – Table 13 shows that the core developers tended to use more work and achievement type language than the other practitioners. These findings are also statistically significant for work ($p < 0.05$), but not so for achievement language ($p > 0.05$). Of the other linguistic dimensions (leisure, social, posemo and negemo), only the leisure (e.g., club, movie, party) category has produced a statistically significant finding ($p < 0.01$) for higher use of this language form for the other practitioners. The Cohen's d effect sizes associated with the differences for work and leisure are both small (0.033 and 0.040, respectively) (Cohen, 1988).

Given these findings, checks are made for differences in the behavioural processes of the two groups of practitioners to ascertain if the nature of the software development activity and/or the specific practitioners involved could have mediated the above results for the ten individual project areas (P1 – P10). A similar pattern of results is found for individualistic and delegation language use across the project areas; however, results for collective language are slightly different, tending to be even across the two groups. While core developers were more collective on some teams (P1 – P3, P6, P7, P9 and P10), the other members were more collective on the remaining teams (i.e., P4, P5 and P8). Another set of Mann-Whitney U tests are conducted for the individual project areas (P1 - P10) for the cognitive dimensions, which has also produced a similar pattern of results as noted for the complete data set. Apart from those working on P1, the core developers for all other teams used consistently higher levels of work and achievement language ($p < 0.01$ is statistically significant for the achieve dimension for higher use of this language form for the other practitioners on P1). Findings for the leisure and social dimensions are also similar to those reported for the entire data set. However, significant differences ($p < 0.05$ and $p < 0.01$) are only observed for the use of leisure utterances on P3 and P8. On the other hand, the other members involved in project areas P2, P6 and P8 expressed significantly higher amounts of negative emotion ($p < 0.01$, $p < 0.01$ and $p < 0.05$, respectively).

In the last stage of the statistical analyses for the linguistic dimensions, checks are made to explore the ways in which core developers expressed behaviours when they were working in more than one team (refer to Table 10 for details). These tests are aimed at understanding if core practitioners' varied their attitudes given the nature of the software tasks they were undertaking. Three of the 13 linguistic dimensions are randomly selected for testing. These included first person pronouns (I), social process words (social), and discrepancy words (discrep); refer to Table 5 for detail on these linguistic dimensions. The distributions for the three selected linguistic dimensions for each of these five core developers in Table 14 are close to normal (only slightly positively skewed), and so checks for differences across the three linguistic dimensions are conducted using *t*-tests. Table 14 shows that the core developers involved in multiple project areas exhibited similar traits across those project areas. Use of first-person pronouns (e.g., I, me, my) was almost identical, while there was also relative consistency in the use of social words (e.g., give, buddy, love) and discrepancy words (e.g., should, would, could). These findings suggest that these core members exhibited quite stable attitudes regardless of the teams in which they were involved (Mehl & Pennebaker, 2003).

Table 14. Results comparing differences in selected language usage for core developers involved in multiple project areas

Contributor	Project areas (Team ID)	t-Test: Two Sample Assuming Unequal Variance (<i>p</i> -value)		
		First-person pronouns	Social process words	Discrepancy words
4661	P1, P2	0.878	0.920	0.888
2419	P1, P2	0.902	0.742	0.685
13722	P3, P10	0.949	0.250	0.089
4674	P3, P5	0.990	0.814	0.244
13664	P6, P7	0.905	0.349	0.603

Overall, these linguistic analysis results revealed that core developers delegated more and used fewer individualistic processes than the less active practitioners. Core developers were also highly task and achievement focused. On the other hand, the less active practitioners used more cognitive processes, engaged more about leisure, and tended to use more collective processes. While these results are insightful, the Cohen's *d* effect sizes associated with the differences in attitudes between core developers and the other practitioners were all small (Cohen, 1988), suggesting that the differences overall, although statistically noteworthy ($p < 0.05$), are of modest practical significance (Kampenes et al., 2007).

There may also be an impact in terms of the analysis method employed. The LIWC tool is applied in a top-down fashion, as its categories of language codes have been pre-determined, and it is quite granular in considering the use of isolated words. It is anticipated that a more exploratory, bottom-up approach focused on phrases might provide different insights into core developers' enacted roles and their knowledge-centred processes. This form of analysis could also triangulate the linguistic findings. Such examinations have led to enhanced understanding of many issues in the software engineering and information systems domains (Sheetz et al., 2009).

The linguistics results reported for core developers and other practitioners attitudes are therefore complemented using directed CA, employing a hybrid classification scheme adapted from related prior work (Henri & Kaye, 1992; Zhu, 1996). As noted in Section 3.4.5, use of the directed approach is appropriate when there is scope to extend or complement existing theories around a phenomenon (Hsieh & Shannon, 2005), and so suited the move to further explore practitioners' messages. The results from this form of analysis are provided in the next section.

4.2.2 Enacted Roles (RQ7)

The results reported above show that Jazz core developers communicated extensively across software tasks and these individuals made the most task (WI) changes (refer to Section 4.1.3). Findings in Section 4.1.3 also show that core developers were not restricted by their formal roles. The linguistic analysis results just reported in Section 4.2.1 have provided further insights into core developers' interaction processes. Mainly, these results revealed that core developers delegated to others more and used higher levels of work and achievement processes. This section extends these SNA and linguistic findings, and considers the nature of core developers' interaction through contextual lenses (Hsieh & Shannon, 2005).

As noted towards the end of the previous section (Section 4.2.1), contextual analysis was conducted using a directed CA approach to answer RQ7 (refer to Chapter 3 for further details). First, all of the messages for three of the ten project areas (those for P1, P7, and P8) were selected (see Table 2). These teams were chosen deliberately as they comprised development efforts aimed at different types of software features (P1 = User Experience tasks, P7 = Coding tasks, and P8 = Project Management tasks). Thus, it was anticipated that potential variations in interaction among those involved in different forms of task would also be revealed by studying these project areas. These three

project areas combined comprised 355 tasks and 1561 messages, with 139 contributors working across the three teams (including 107 distinct members) (refer to Table 2).

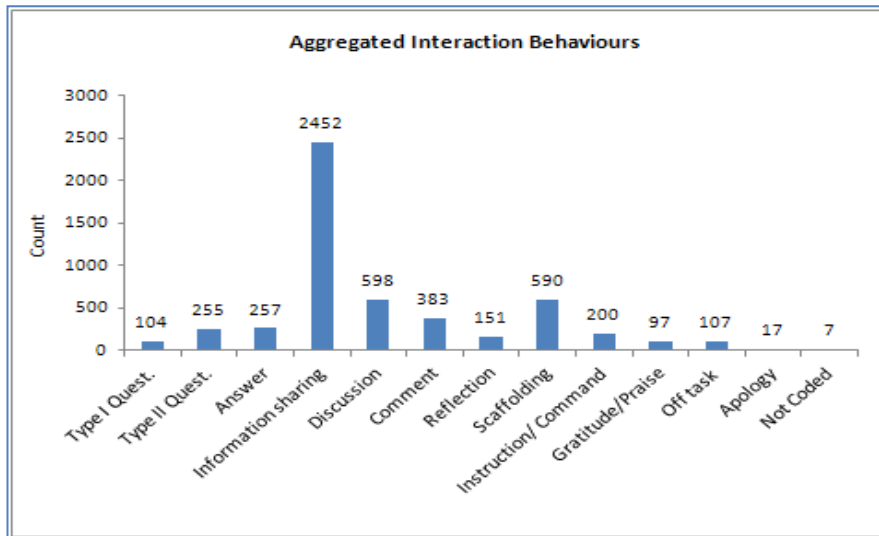
All of the 1561 messages were coded using the direct CA approach outlined in Section 3.4.5.8. As noted, the coding process employed a multiphase approach, including creating the protocol from previous studies used for measuring interaction, checking the suitability of the protocol for analysing software developers' interactions by studying a small sample of IBM Rational Jazz teams' messages and extending the protocol accordingly, recoding all of the messages with this extended protocol, and finally, checking that the codes were reliably obtained. These steps form the process for conducting a reliable and valid content analysis study (Hsieh & Shannon, 2005; Jonassen & Kwon, 2001; Rourke & Anderson, 2004) (refer to Chapter 3 for further details).

From the total 1561 messages that were coded, 5218 utterances were recorded for the three teams (P1 = 1165 codes, P7 = 1770 codes and P8 = 2283 codes). Table 2 shows that the three project areas selected for coding were of varying duration. There were also variances in the numbers of tasks, messages and contributors in these project areas, as is evident in Table 2. Overall, Table 15 shows that, typically, the highest number of messages (8.5) was exchanged on user experience related tasks (P1), while coding activities (P7) resulted in the second highest number of messages (5.4) on average for each task, and practitioners working on the project management tasks (P8) exchanged the least number of messages (2.9) on average. Measures for the average number of tasks for each contributor were also different. On average, those working on the coding team (P7) addressed the highest number of tasks (5.7); while members worked on fewer tasks (1.6) in the user experience project team (P8) (refer to Table 15). Table 15 further shows that those working on the coding team (P7) also communicated the most mean messages (30.6), while the user experience project team contributors (those of P1) were next in line, sharing 13.9 messages on average, and those on the project management team (P8) were the least 'vocal' – these practitioners exchanged 6.8 messages on average. Although contributors to the project management team (P8) typically communicated the least around software tasks (2.9 messages on average), and this was similar in terms of the average messages for each contributor (2.3), on average these individuals said more in each message (3.7 utterances compared to 3.6 and 2.5 for practitioners of coding (P7) and user experience (P1) tasks, respectively) (refer to the Codes/Message column in Table 15).

Table 15. Mean project area measures for messages, tasks, contributors and codes

Team ID	Messages/ Task	Tasks/ Contributor	Messages/ Contributor	Codes/ Message
P1	8.5	1.6	13.9	2.5
P7	5.4	5.7	30.6	3.6
P8	2.9	2.3	6.8	3.7

Figure 24 shows the distribution of the aggregated interaction behaviours (from the 5218 codes that were derived) that occurred during all three project teams (P1, P7 and P8). In Figure 24 it is noted that Information sharing (2452 codes), Discussion (598 codes), Scaffolding (590 codes) and Comments (383 codes) were the most dominant behaviours evident during Jazz practitioners' discourses. Additionally, Apology type communication (17 codes) was rarely observed, and only a few utterances were not matched to a category (Not Coded = 7 codes). Figure 24 shows that Type I Questions (104 codes), Gratitude (97 codes) and Off task (107 codes) utterances recorded low usage and were relatively even in number. A similar pattern is evident in Figure 24 for Type II Questions (255 codes), Answers (257 codes) and Instructions (200 codes). The number of codes for Reflection (151 codes) was slightly lower than that for Instructions (refer to Table 6 for details). Please refer to Appendix III and Appendix IV for additional visualisations of the codes across the three project areas (P1 = User Experience (UE), P7 = Coding (Code), and P8 = Project Management (PM)).

**Figure 24. Behaviour category (utterances) and number of occurrences for P1, P7 and P8**

In alignment with the focus of the analysis in this phase – to explore the roles that are enacted by core developers – all of the 2191 codes that were contributed by the six core developers of the three project areas (P1, P7 and P8) are extracted and analysed. In total, core developers working on P1 (UE) communicated 648 codes, the core developers

working on P8 (Code) communicated 1245 codes and core developers working on P8 (PM) communicated 298 codes. In order to assess these codes against those of the core developers overall project areas' (or teams') utterances (as was done in Section 4.1.3), Figure 25 shows the percentages of overall project teams' interaction behaviours for the individual core developers in the user experience (P1), coding (P7) and project management (P8) project areas. (Note that the Not Coded category is not included in Figure 25 as altogether only four (0.18%) core developer utterances were matched to that category, refer to Table 6).

For the user experience project area (P1 or UE), Figure 25 illustrates that contributor 4661 articulated 75%, 55% and 51% of the team's Instructions, Off task communication and Reflective utterances, respectively. This contributor (4661) also communicated nearly a half of the Type II Questions (enquiries), Discussions (ideas) and Scaffolding (suggestions and guidance) on the user experience project area (UE). Additionally, contributor 4661 expressed 41% of the team's Gratitude, contributed a third of the team's Information and Comments (debates) and communicated a quarter of the team's Type I Questions and Answers (refer to Figure 25). On the same project area (UE) Figure 25 shows that contributor 2419 was involved in nearly a third of the team's Type I Questions, shared a quarter of the team's Information, provided nearly a fifth of the team's Type II Questions and Answers, and offered 11% of the team's Instructions.

The core developers' dominance is maintained on the coding project area (P7 or Code) where Figure 25 shows that contributor 12972 contributed over half of the team's Reflection (evaluation and self-appraisal) type utterances and Discussions (ideas). Figure 25 reveals that this contributor (12972) also communicated half of the team's Comments (debates), Information, Off task utterances, and Type II Questions (enquiries). Contributor 12972 provided 43% of the coding team's (Code) Scaffolding communications (suggestions) and Type I Questions, 37% of the team's Answers, and nearly a fifth of the team's Instructions (refer to Figure 25). On this project area (Code), Figure 25 illustrates that contributor 13644 provided half of the team's Instructions, a third of the team's Scaffolding utterances (suggestions) and Answers, a quarter of the team's Off task communications, and a fifth of the team's Information and Discussions (ideas). This contributor (13644) also provided 16% of the team's Comments (debates) and 14% of the team's Reflections.

Figure 25 shows that the core developers' influence observed on the user experience (UE) and coding (Code) project areas was less evident on the project management project area (PM). On the project management project area (PM) contributor 12702 provided close to a quarter of the team's Instructions, while practitioner 2102 contributed just under half of this measure on the team (PM). Similarly, Figure 25 demonstrates that measures for Information sharing, Comments (judgemental or debate type utterances) and Answers to information seeking questions were lower for 12702 and 2102. In fact, these contributors scored less than 10% of the team measures for all but one of the other coding categories (Question, Discussion, Reflection, Scaffolding, Gratitude, Off-task communication) (refer to Figure 25). These findings are not unexpected for the core developers that were selected for P8 (PM), particularly because none of the members that were selected in the core members' group met the original selection criterion (both members had a density measure lower than the baseline of ≥ 0.33 that was initially set (Crowston et al., 2006)), refer to Section 4.1.3. Given that this project area comprised 90 different contributors, however (compared to 33 contributors on P1 (UE) and 16 contributors on P7 (Code)), these measures are still quite revealing (see Table 2 for details).

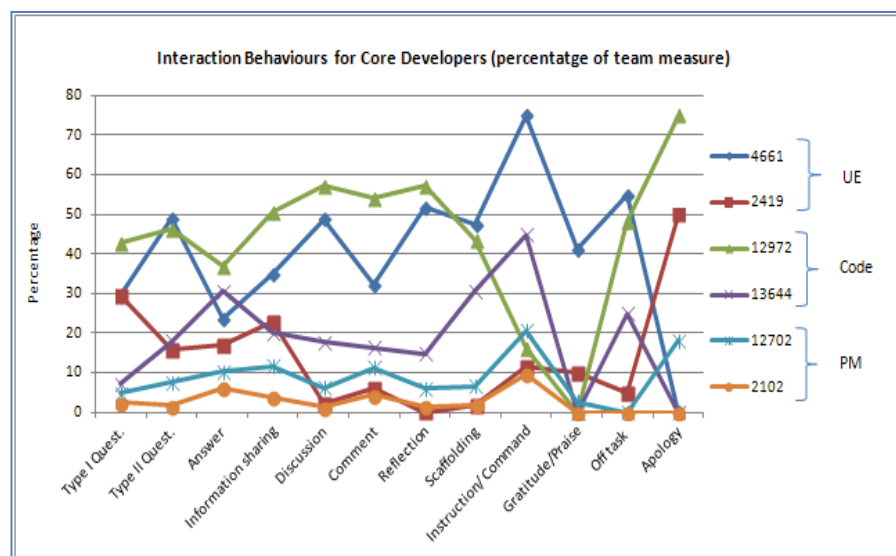


Figure 25. Percentages of overall team interaction (utterances) behaviours for the core developers

Overall, the core developers provided 42% (2191 codes) of their teams' utterances (see Appendix V for core developers' combined interaction patterns for the three project areas (P1, P7 and P8)). These utterances were related to interpersonal, intrapersonal and organisational forms of communications. This finding is revealing when considering that these core developers comprised just 5.6% (or 6 members) of their teams' 107

distinct members. These figures are particularly surprising when allowing for what would be a ‘typical’ contribution of knowledge if codes were distributed evenly across the 107 contributors to the three project areas (P1 (UE), P7 (Code) and P8 (PM)) – taking this approach a team member would typically have contributed around 48.77 codes, or 0.93% of their teams’ overall utterances. Taking this assessment into consideration, core developers shared over 45 times (or each core developer shared 7.5 times) the knowledge of the average practitioner on their teams.

These differences are further illustrated in Figure 26 which provides visualisations of the interaction behaviours for core developers and the other team members of the user experience project area (P1) (refer to Appendix VI and Appendix VII for similar visualisations for the coding (P7) and project management (P8) project areas). In Figure 26 it is shown that, overall, core developers dominated most of their team’s interactions. In fact, these practitioners were working in a team comprising 33 members, which included 18 programmers, 11 team leads, 2 project managers, 1 admin, 1 multiple roles. The core developers of the user experience project area were one team lead (of the 11 team leads) and one programmer (of 18 programmers) (refer to Table 16 for P1 (UE), P7 (Code) and P8 (PM) core developers’ formal role assignment information). This evidence is revealing, particularly given that the UE team also comprised two project managers (noted earlier).

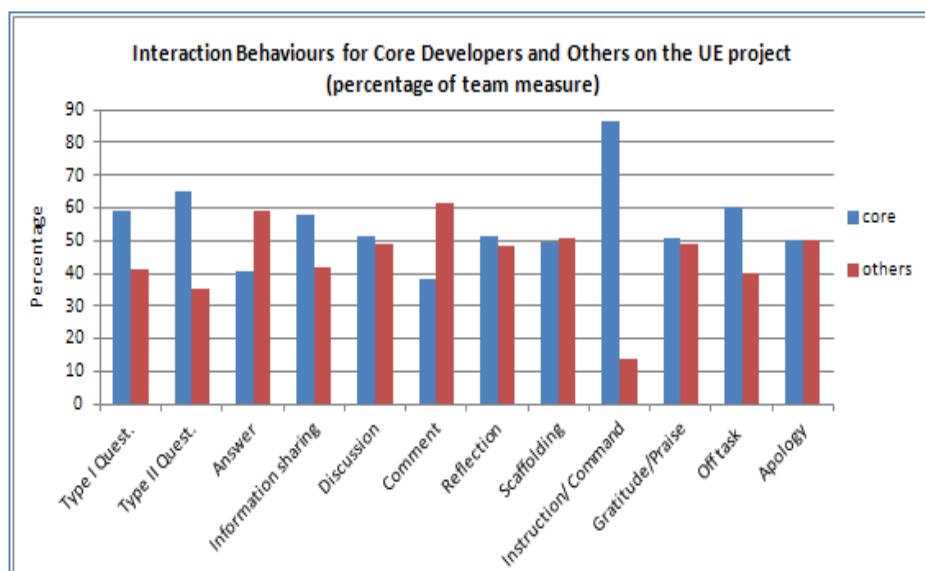


Figure 26. Summary of project interaction (utterances) for the core developers and others (for the UE team (P1))

Table 16. P1 (UE), P7 (Code) and P8 (PM) core developers' formal roles

Team ID	Project Member	Role
UE (P1)	4661	Team lead
	2419	Programmer
Code (P7)	12972	Programmer
	13664	Team lead
PM (P8)	12702	Project manager
	2102	Team lead

Pearson Chi-square tests are conducted to ascertain whether the differences observed in the visualisations shown in Figure 26 (and those in Appendix VI and Appendix VII) are statistically significant. This statistical procedure is viewed as appropriate when the distributions comprise frequency data, as is the case for the codes that were obtained for P1, P7 and P8 through the directed CA process (Sharp, 1979). Additionally, given that the data revealed is categorical (refer to Table 6), the Chi-square test is the statistical procedure of choice. Further, with the exception of the Not Coded category (only seven codes were recorded for this category – refer to Figure 24), all the data samples (for the other 12 categories – refer to Figure 24) comprised a sample size that was substantially more than ten (the assumption for utilising a Chi-square test) (Sharp, 1979).

Thus, three Chi-square tests are conducted. First, all of the 5218 codes are separated and then aggregated along the 12 coding categories for the two groups (core developers and others). In this aggregation, core developers contributed 2191 codes and the other members contributed 3027 codes. In line with the overall pattern (wherein the others' codes combined were more than those contributed by core developers), it is noted that the other members contributed more than 50% of the codes for all the coded categories except Instruction/ Command (other members contributed 37.5% of the overall instructions) and Off task utterances (this category was even at 50% each). The first Chi-square test is conducted to check these two groups (core developers and others) for differences, and the result is statistically significant, $X^2(12) = 74.383$, $p < 0.001$. The effect size for this finding, Cramer's V, is small, 0.119 (Cohen, 1988). This result suggests that, overall, the other members contributed significantly more interactions than the core developers (with the exception of Instructions/ Command and Off task utterances), though, this difference is of modest practical significance (Kampenes et al., 2007).

Given the pattern that was noted earlier for the core developers of the project management project area (PM or P8) (i.e., these members contributed significantly

lower amounts of interactions than those working on the user experience (P1) and coding (P7) project areas, they were selected in the core developer cluster having failed the original selection criteria (both members had a density measure lower than the baseline of ≥ 0.33 that was initially set (Crowston et al., 2006)), and their project area comprised 90 members – the largest number for all the teams (P1 – P10), it is suspected that the measures (codes obtained) for the core developers of the project management project area (P8) are mediating the overall results of the first Chi-square test (discussed in the previous paragraph). Thus, two additional Chi-square tests are conducted to ascertain whether a difference is present in the way core developers contributed their knowledge on the user experience (P1) and coding (P7) project areas respectively. For the user experience project area (P1), the result shows that core developers contributed more than 50% of the codes for eight of the 12 interaction categories that are examined (other members contributed 58.2% of the Answers and 61.8% of the Comments; and two categories (Scaffolding and Apology) were even – refer to Figure 26 for visualisation of codes). The Chi-square result is statistically significant, $X^2(12) = 60.813$, $p < 0.001$. The effect size, although small (Cramer's V, 0.228) (Cohen, 1988), is larger (and so of greater practical significance) than those seen for the overall Chi-square result above (0.119).

A similar pattern of results is revealed for the coding project area (P7). For this team core developers contributed more than 50% of the codes for 10 of the 12 interaction categories (Gratitude being the exception), and codes were even for Type I Questions among core developers and other members (both groups contributed 50% of this category – refer to Appendix VI for visualisation). The Chi-square result is statistically significant, $X^2(12) = 32.270$, $p < 0.01$. The effect size for this finding, Cramer's V, is small (but higher than the overall result above as well), 0.135 (Cohen, 1988).

Additionally, in line with the pattern noted in the user experience team (P1) above, the core developers working on the coding team (P7) were also one programmer and one team lead (refer to Table 16 for details). In Table 2 it is shown that P7 comprised of 16 members; including 6 programmers, 7 team leads, 1 project manager, 1 admin, 1 multiple roles.

These directed CA results show that core developers were major sources of their teams' knowledge and these members also contributed to their teams' social climate.

Additionally, core developers offered most of their teams directives (Instructions) and guidance (Scaffolding). These forms of interactions are aligned to both task based and

social roles (Benne & Sheats, 1948). The statistical analysis also confirmed that core developers expressed significantly different knowledge behaviours when compared to the other practitioners of their teams. However, while these results and those in Section 4.2.1 establish that core developers exhibited significantly different attitudes and behaviours to the regular developers during team work, these results only reflect a static view of the software project. This is in line with the objective to establish whether core developers' attitudes differ to those of regular team members, and examining the roles these practitioners enacted. Questions related to how core developers share knowledge over the course of their project, the initial team arrangements that cause core developers to become hubs in their teams and how the behaviours and traits these practitioners exhibited are linked to their involvement in task performance have not been answered. Such explorations could provide understandings for the peculiarities of globally distributed agile software team dynamics, may inform appropriate team configurations, and may enable the early identification of 'software gems' – exceptional practitioners in terms of both task and team performance. These questions may only be answered through longitudinal examinations. The next section adopts this approach in the analysis of core developers' artefacts, towards delivering answers for the preceding questions.

4.3 Phase 3 – Linguistic Analysis and Directed CA (Longitudinal Analyses)

Results in Section 4.1 revealed that few members dominated project communication and these members were also integrally involved in their teams' software development portfolio (Reagans & Zuckerman, 2001). Additionally, the results in this section (Section 4.1) revealed that these core developers were not restricted by their formal role assignment (Datta et al., 2010). As an initial step towards understanding the reason for these members' pronounced communication and task performance, Section 4.2 compared the attitudes of these core developers to those of their less active counterparts, and investigated the roles that were enacted by core developers during their project. The drive to understand core developers' attitudes and their enacted roles is invaluable given that these practitioners occupy the centre of their teams' coordination action, they are seen as the project's leaders (whether or not they are the formal leaders (Hinds & McGrath, 2006)), and they coordinate information flow and knowledge within their teams (Leavitt, 1951). Accordingly, the nature of core developers' attitudes, and the role these members enact during their project development, are likely to be determinants of their teams' success.

For instance, these members are likely to express happiness and satisfaction in their communication if they are fulfilled, while the opposite may be observed if they are dissatisfied (Stone & Pennebaker, 2002). During core developers' periods of dissatisfaction, team communication may not be properly facilitated. The more reliant team members may also find it unsettling to solicit help from core developers when they exhibit negative attitudes. In fact, for globally distributed software developments, negative and cynical team behaviours may have an overall generally negative impact on team harmony and cohesion (Chang et al., 2013; Denning, 2012). The negative effects of such behaviours may be particularly evident given that there are often reduced opportunities for cooperation during global developments (Serce et al., 2009), and the collaborative technologies that are commonly used in these settings generally offer only limited amounts of social presence (Cummings, 2004; Herbsleb & Grinter, 1999). Thus, reduced levels of team harmony and cohesion may affect global teams' performance (Espinosa et al., 2007). The opposite is likely to occur in more optimistic and social environments where global teams share a single vision.

Similarly, the roles that core developers enact in their teams are likely to determine their teams' effectiveness at managing interpersonal communication, conflicts and software quality (Hayes Huffman, 2003). Helpful and supportive behaviours (personal and social roles) and task-concerned behaviours (task roles) are the desired roles for maintaining task performance (Benne & Sheats, 1948), whereas, excess debate and conflict-centred behaviours (individualistic roles) have a negative effect on task performance (Andre et al., 2011; Chang et al., 2013). Core developers' expression of social and positive behaviours may be especially necessary during times of high intensity and stressful team work – providing encouragement and support for the less strong team members – whereas, these core developers' expression of task-driven attitudes may be most effective during actual task analysis and brainstorming stages, or when the less active members display reduced focus on task performance. Should these members exhibit moderate levels of individualistic roles, this may also be useful for maintaining high team standards through critical and constructive debates. The results in this section are aimed at examining these issues further, as the Jazz software project change from phase to phase.

In alignment with the process for normalisation of the project phases and artefacts described in Section 4.1, core developers' artefacts are separated according to the four project phases (start, early-mid, late-mid and end). Table 17 provides a summary of the

messages contributed by the core developers over the four phases of their project (refer to Appendix VIII for additional descriptive statistics). As noted in Section 4.2.1, in total, 2565 messages (shown in Table 17) were contributed by the core developers (of the total 5563 messages noted in Table 2). These practitioners typically communicated most in the early and middle phases of their project (see the measures for P1, P2, P5, P7 and P8 in Table 17). Previously, it was revealed that, overall, Jazz teams communicated most in the first and last phases of their project (Licorish & MacDonell, 2012) (these results are also presented in Section 4.1.1 above), suggesting that the less active developers communicated more towards project completion. This finding was also noted by Cataldo and Herbsleb (2008), who discovered that technical dependencies resulted in increased levels of communication for some of the less active developers at various times of the project.

In order to verify whether there were significant differences in core developers' contribution of messages over the four project phases, first, checks for the normality of the data distributions are conducted using Shapiro-Wilks tests. These tests confirm that the data did not violate the normality assumption for any of the four project phases. Therefore, a two-way ANOVA test for significant differences is conducted to examine core developers' contribution of messages over the course of their project. There is homogeneity of variance between the four phases as assessed by Levene's test for equality of error variances. The ANOVA test uncover that although core developers tended to communicate more in the first three project phases, these differences are not statistically significant, $F(3, 36) = 0.191$, $P > 0.05$.

Table 17. Numbers of messages communicated by core developers

Team ID	Phase				Σ
	start (%)	early-mid (%)	late-mid (%)	end (%)	
P1	51 (19.8)	96 (37.2)	55 (21.3)	56 (21.7)	258
P2	138 (26.7)	184 (35.6)	106 (20.5)	89 (17.2)	517
P3	25 (52.1)	11 (22.9)	7 (14.6)	5 (10.4)	48
P4	83 (28.2)	73 (24.8)	72 (24.5)	66 (22.5)	294
P5	38 (20.5)	28 (15.1)	73 (39.5)	46 (24.9)	185
P6	77 (19.8)	96 (24.7)	93 (23.9)	123 (31.6)	389
P7	75 (23.7)	82 (26.0)	89 (28.2)	70 (22.2)	316
P8	22 (16.3)	28 (20.7)	52 (38.5)	33 (24.4)	135
P9	42 (36.2)	33 (28.5)	21 (18.1)	20 (17.2)	116
P10	106 (34.5)	72 (23.5)	63 (20.5)	66 (21.5)	307
Σ	657 (25.6)	703 (27.4)	631 (24.6)	574 (22.4)	2565

As noted earlier, whereas in Section 4.2 core developers' attitudes and their enacted roles were examined using a project snapshot (static analysis) approach, this section presents longitudinal analyses of core developers' attitudes, knowledge sharing behaviours and task performance over their project. Messages (refer to Table 17) are examined to study core developers' attitudes and knowledge sharing behaviours, and change log activities are investigated to study their task performance (refer to Section 3.4 for further details).

Section 4.3.1 reports the exploration of the way core developers express attitudes as their project progress, through the use of linguistic analysis techniques (as used in Section 4.2.1). This aspect of the analysis is aimed at answering RQ8 (Do core developers' attitudes change as their project progress?). Section 4.3.2 then provides the directed CA (as used in Section 4.2.2) results pertaining to the way core developers share knowledge over their project, thus answering RQ9 (How do core developers share knowledge over the course their project?) and RQ10 (What initial team arrangements lead to core developers becoming hubs in their teams?). RQ11 (How do core developers contribute to task performance over their project?) is then answered by the task changes results presented in Section 4.3.3. The results comparing core developers' attitudes with their task performance are presented in Section 4.3.4, towards answering RQ12 (Are core developers' contributions to task performance linked to their attitudes?). Finally, directed CA and task performance results are outlined in Section 4.3.5, and are aimed at answering RQ13 (Are core developers' contributions to task performance linked to their contribution of knowledge?).

4.3.1 Attitudes (RQ8)

In order to answer RQ8 core developers' messages (the 2565 total messages) are analysed according to the 13 linguistic dimensions in Table 5. Each of these individual distributions are checked for normality (Brooks et al., 1994) over the four project phases (start, early-mid, late-mid, and end) of the ten project areas using the Shapiro-Wilks test. Homogeneity of variance over the different project phases is found using Levene's test for equality of error variances for all of the 13 linguistic dimensions. Results for the Shapiro-Wilks tests for ten of the thirteen linguistic dimensions are found to be normally distributed across the four project phases (Onwuegbuzie & Danlel, 2002). Closer examination of the standardised skewness coefficient (i.e., the skewness value divided by its standard error) and standardised kurtosis coefficient (i.e., the kurtosis

value divided by its standard error) for these ten linguistic dimensions has further established that these distributions conformed to the normality assumption. Measures for collective (we), leisure and positive emotion (posemo) language have failed the Shapiro-Wilks normality test. For these three dimensions (collective, leisure and positive emotion language – refer to Table 5), the standardised skewness and kurtosis coefficients are also outside the boundaries of normally distributed data (i.e., -3 to +3) (Onwuegbuzie & Danel, 2002).

Therefore, two-way ANOVA tests are conducted to check for differences in mean linguistic scores over the four project phases for the ten linguistic dimensions with normal distributions, and the equivalent non-parametric Kruskal-Wallis tests are conducted for the other three non-normally distributed linguistic dimensions just mentioned (Onwuegbuzie & Danel, 2002) (refer to Table 18 for the descriptive statistics concerning core developers' linguistic usage over their project).

Overall, although there were differences in the way core developers used the different language dimensions over time, these differences are not statistically significant ($p > 0.05$). For instance, it is revealed that core developers used slightly lower levels of individualistic language at the start of their project (mean 6.8, median 6.6, Std dev 4.5), and use of this language form increased slightly as the project progressed to completion (mean 7.9, median 7.2, Std dev 3.5). Core developers were less collective in the early phases of their project (mean 2.9, median 3.0, Std dev 1.7), and these practitioners tended to be most collective towards project completion (mean 4.1, median 2.9, Std dev 3.4). Core developers were also highly work focused (frequently using words like “feedback”, “goal” and “delegate”) towards the end of their project (mean 13.5, median 11.4, Std dev 7.3). Additionally, core developers used a high amount of social language (e.g., give, buddy, love) throughout their project, but became less social as their project progressed (means: start 14.8, early-mid 12.9, late-mid 12.7, and end 12.7). Finally, while negative language use (e.g., afraid, hate, dislike) was low overall for core developers (mean 3.5, median 3.3, Std dev 2.8), these practitioners expressed negative emotion mostly towards project completion (mean 4.8, median 3.9, Std dev 4.9) (refer to Table 18 for further details).

Table 18. Descriptive statistics for core developers' linguistic measures across the project phases

Abbrev.	Mean				Median				SD				SK				KS			
	start	early-mid	late-mid	end	start	early-mid	late-mid	end	start	early-mid	late-mid	end	start	early-mid	late-mid	end	start	early-mid	late-mid	end
I	6.8	7.6	7.5	7.9	6.6	9.1	8.9	7.2	4.5	4.7	4.4	3.5	0.3	-0.4	-0.3	-0.2	-0.2	-1.6	-1.6	-0.7
we	2.9	2.8	2.4	4.1	3.0	2.3	2.1	2.9	1.7	2.7	1.3	3.4	-0.0	1.6	0.2	2.0	-0.7	2.8	0.6	4.5
you	2.9	3.4	3.5	2.4	2.8	3.5	3.0	2.6	1.0	1.4	1.7	1.9	0.2	0.3	0.8	0.1	0.6	-1.4	0.7	-1.3
insight	5.7	6.7	6.4	5.7	5.1	5.3	6.2	5.4	3.4	4.2	2.2	1.4	1.5	1.0	0.6	0.2	2.9	0.2	0.3	-1.2
discrep	6.4	6.5	5.7	5.8	6.9	5.6	5.9	5.1	2.4	3.0	3.2	4.6	-0.3	0.9	-0.2	1.5	-0.1	0.1	-1.5	3.1
tentat	5.7	5.6	4.9	5.4	5.9	5.2	4.2	5.0	2.7	2.8	3.5	3.1	-0.2	0.6	0.9	1.4	-1.6	-1.4	-0.1	2.4
certain	2.3	1.9	2.6	2.4	2.8	1.9	2.5	2.4	1.3	0.8	1.5	1.7	-0.5	-0.3	0.3	0.5	-1.5	-1.5	-1.1	-1.1
work	11.8	11.0	14.0	13.5	10.5	10.2	14.4	11.4	3.1	3.9	2.1	7.3	0.5	0.6	0.3	1.6	-0.8	-0.2	0.8	3.1
achieve	10.5	10.2	10.5	10.7	10.6	9.7	10.8	9.4	3.7	4.5	3.3	5.3	-0.3	0.6	-0.2	0.3	-0.9	-0.5	-1.4	-1.3
leisure	3.1	2.6	3.3	2.7	3.1	2.1	2.6	1.7	1.2	2.1	1.7	2.8	0.0	1.3	1.1	2.0	-1.3	1.0	0.3	4.6
social	14.8	12.9	12.7	12.7	14.9	12.3	13.3	13.2	3.1	3.5	4.3	4.2	-0.4	0.3	-0.5	-0.8	-0.3	-0.2	-0.9	0.3
posemo	15.7	17.9	18.4	17.0	9.9	11.8	13.6	12.3	15.2	14.8	12.8	15.2	1.7	1.2	1.1	1.4	2.1	-0.0	-0.3	1.1
negemo	3.1	2.7	3.6	4.8	2.8	2.9	3.4	3.9	1.8	1.5	1.6	4.9	0.8	-0.1	0.9	2.0	-0.2	-0.9	1.5	4.8

Notes: SD = Standard Deviation, SK = Skewness, KS = Kurtosis

Generally, the linguistic analysis findings show that core developers' attitudes did not change excessively over the course of their project. Additionally, taken as a whole, the results show that when there were some levels of changes in core developers' attitudes, these changes were mixed – some attitude changes are generally desirable, while others are unfavourable for teamwork (Denning, 2012). For instance, evidence for the way core developers became more collective and work-focused as their project progressed is a positive sign for team performance (Licorish & MacDonell, 2012). On the other hand, the higher incidence of individualistic and negative attitudes towards project completion is a less attractive indicator (Benne & Sheats, 1948; Chang et al., 2013). These results are triangulated through contextual analysis techniques in the next section (Section 4.3.2) to unearth further details around these changes.

4.3.2 Knowledge Sharing (RQ9) and Becoming Team Hubs (RQ10)

In order to answer RQ9 and RQ10 contextual analyses were conducted using directed CA to study core developers' knowledge sharing behaviours. These results increment those that were revealed through linguistic analysis techniques in the preceding section (Section 4.3.1). As with Section 4.2.2, first, core developers' messages were coded. Core developers' knowledge sharing behaviours were then examined from these codes, though appropriate statistical techniques.

Of the 1581 messages that were coded in Section 4.2.2, 709 of these messages were contributed by the core developers of the three project areas (refer to Table 17). As noted in Section 4.2.2, these three project areas (P1, P7 and P8) were deliberately selected as they represented different task portfolios, and so, it was anticipated that such insights would reveal differences among teams solving different types of tasks (Licorish & MacDonell, 2012). Thus, the contextual analysis that is conducted here could potentially triangulate prior results. From the 709 messages, 2191 utterances (P1 = 648 codes, P7 = 1245 codes, and P8 = 298 codes) were recorded for the core developers.

In Figure 27 and Figure 28 percentages are used to represent the differences in core developers' knowledge sharing contributions to their teams' knowledge pools during the four different phases of their project (refer to Table 19 for counts of core developers utterances). Figure 27 (a) provides aggregated summary percentages of the core developers' knowledge sharing interactions during their project. Here it is observed that Information Sharing, Discussion, Scaffolding, Comments and Instructions dominated core developers' discourses. This pattern of results is also maintained for core

developers' interactions on the individual project areas (P1, P7 and P8) in Appendix IX (a). Figure 27 (b) shows that core developers contributed 60% of their knowledge during the middle stages of their project. In reviewing the details of core developers' interactions for the individual project areas (refer to Appendix IX (b)), this higher contribution of knowledge sharing in the middle project phases is also maintained. A Chi-square test (refer to Section 4.2.2 for discussions around appropriate use of Pearson Chi-square tests) confirm that there were significant differences in the levels of contribution of core developers over the different project phases, $X^2(36) = 63.237$, $p = 0.003$. The effect size for this finding, Cramer's V , is small, 0.168 (Cohen, 1988).

To probe these overall results further, the nature of core developers' knowledge sharing interactions during their project is considered in detail by examining the graphs in Figure 28 (a-d). The directed CA results reveal that although core developers did not ask many questions overall (refer to Figure 27 (a)), this type of utterance increased towards project completion: 34.6% and 30.6% of Type I and II Questions were asked in the last project phase (refer to Figure 28 (a)). Figure 28 (a) also shows that the majority of core developers' Answers to their teams' questions were provided in the middle phases (34.0% and 33.2% in early-mid and late-mid phases, respectively). Figure 28 (b) further shows that core developers shared most Information (33.8%), expressed more Ideas (30.9%) and offered the most Suggestions (31.4%) to their teams during the later middle phase of their project. The overall trend of the graphs shown in Figure 28 (b) indicates that core developers were most engaged in the middle (early-mid and late-mid) phases of their project. In looking at the Comments in Figure 28 (c) it is noted that there was a high contribution of this form of expression by core developers in the second project phase (more than 35%), but use of this language form remained relatively stable in the last two project phases (23.1% recorded in both phases). Figure 28 (c) shows that core developers were most Reflective towards project completion (32.2% of this language form was used in the end phase), and although core developers did not contribute a large amount of Off task communication or Gratitude (see Figure 27 (a)), these forms of utterances were also used most in the middle phases by these practitioners (see Figure 28 (d)). The frequencies for Apology and Not Coded utterances were not plotted in Figure 28 (as were shown in Figure 27 (a)), since only four and six codes (of the 2191 codes) were recorded to these categories, respectively.

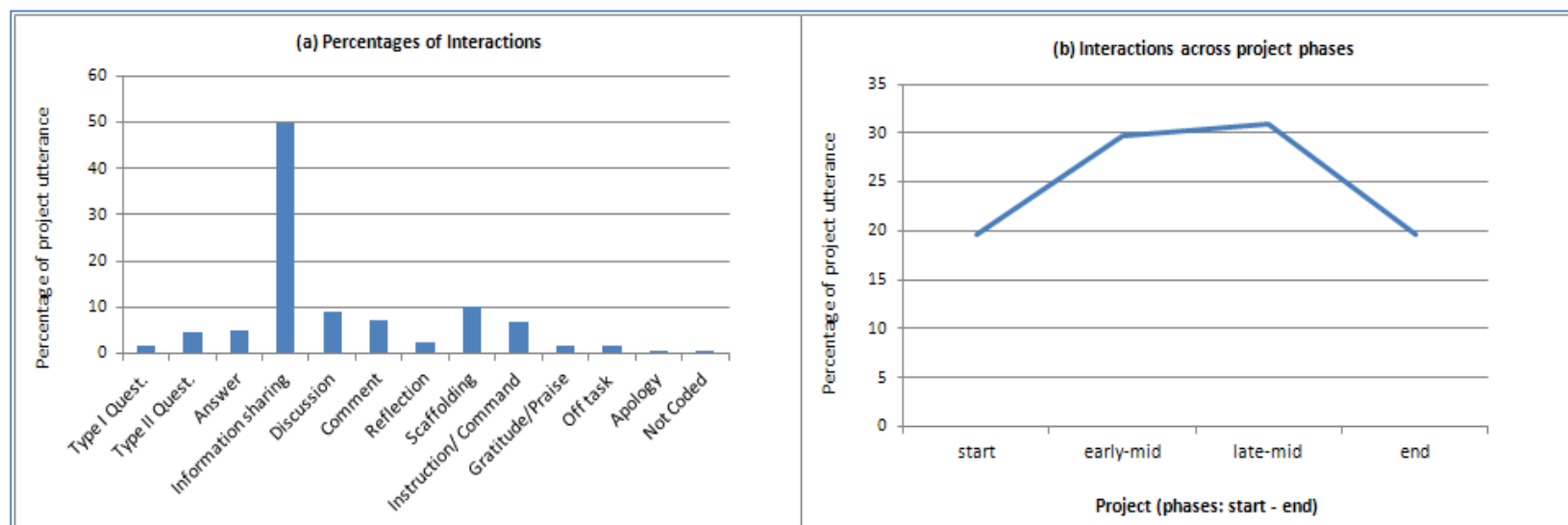


Figure 27. Aggregated interactions (utterances) for core developers

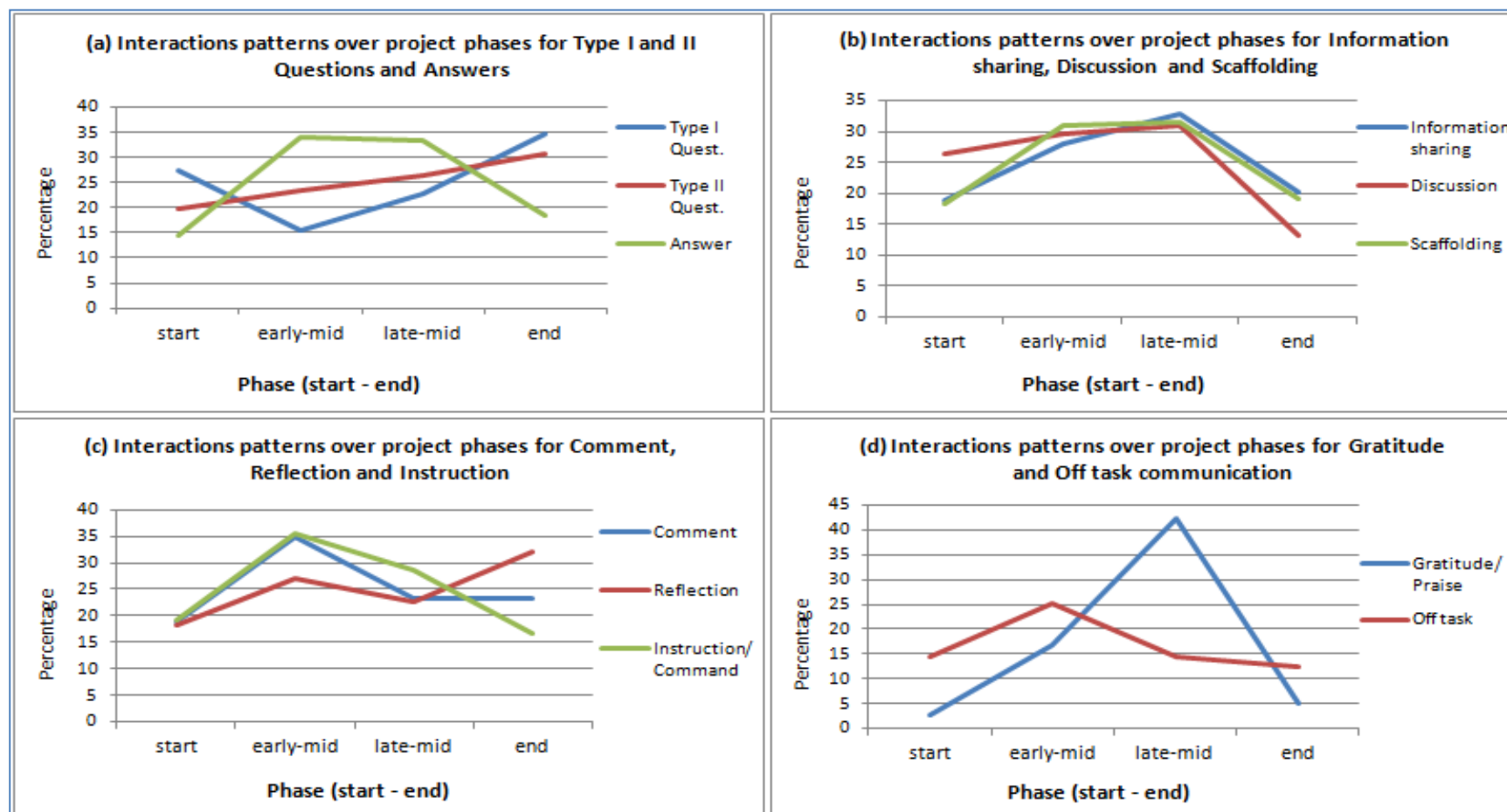


Figure 28. Detailed interactions (utterances) of core developers over project phases

Table 19. Counts of core developers' interactions (utterances)

Category	Phase				Σ
	start	early-mid	late-mid	end	
Type I Quest.	9	8	8	12	37
Type II Quest.	20	25	26	27	98
Answer	15	31	29	19	94
Information sharing	214	303	328	206	1051
Discussion	59	63	56	22	200
Comment	39	69	36	29	173
Reflection	17	20	14	9	60
Scaffolding	52	86	80	45	263
Instruction/ Command	30	52	26	17	125
Gratitude/Praise	2	13	8	4	27
Off task	13	19	13	8	53
Apology	0	1	3	2	6
Not Coded	3	0	0	1	4
Σ	473	692	630	396	2191

The directed CA results above show that core developers established pronounced knowledge sharing roles in their teams during the early project phase, however, these individuals contributed the most of their knowledge in the early-mid and late-mid stages of their project. In contrast, core developers were least visible in their teams' knowledge networks towards project completion (in the end phase). These differences were shown to be statistically significant (although the effect size was modest); slightly divergent to the results revealed through linguistic analysis techniques (refer to Section 4.3.1). In general, these results confirm the importance of temporal analysis for studying the changes in teams' dynamics as software projects progress (Hinds & McGrath, 2006). However, while the results in this section do support the view that core developers evolved and behaved differently over their project, and so, provides answers for *how* core developers contribute to their teams' project knowledge dynamics from project phase to project phase, it is still unclear *why* these patterns exist. Of further relevance are how core developers are involved with task performance over their project (considered in the next section – Section 4.3.3), and how these involvements relate to their attitudes and knowledge sharing behaviours (considered in Section 4.3.4 and Section 4.3.5 respectively). These insights provide additional pointers towards unfolding the reasons for (*why*) the particular patterns that are observed in this section.

4.3.3 Task Performance (RQ11)

In the preceding sections (Section 4.1 and Section 4.2) it was revealed that core developers were very active, and these members made the most changes to their teams' tasks. In Section 4.3.1 it was observed that although core developers' expression of attitudes over the course of their project did not differ in a statistically significant manner, some specific attitudes did appear to be more pronounced during certain project phases. These results were extended in Section 4.3.2, where it was discovered that core developers expressed significantly different knowledge sharing behaviours over the course of their project. In this section a detailed analysis is conducted to unearth how core developers contributed to task performance as their project progressed. This analysis is aimed at answering RQ11.

Table 20 provides a summary of the task change data for core developers, and shows that these members tended to make the most changes to their teams' tasks during the middle stages of their project. These individuals made as many as 33% of their task (WI) changes in the early-mid phase for P2, and 47% of their task changes in the late-mid phase for P8 (see Table 20 for overall means which also maintain this pattern). The higher levels of task changes in these phases coincide somewhat with the higher numbers of messages communicated and knowledge sharing behaviours expressed by these individuals, as noted in Table 17 and Table 19. Note also that a small, positive correlation was previously observed between the number of messages communicated and the number of task changes made by core developers (Licorish & MacDonell, 2013c).

In order to verify whether there are significant differences in core developers' contribution of task (WI) changes over the four project phases, first, checks for the normality of the data are conducted using Shapiro-Wilks tests. These tests confirm that core developers' task change measures did not violate the normality assumption for any of the four project phases. Therefore, a two-way ANOVA test is conducted to examine core developers' contribution of task changes over the course of their project for significant differences. There is homogeneity of variance between core developers' task changes over the four project phases as assessed by Levene's test for equality of error variances. The ANOVA test reveal a statistically significant difference $F(3, 36) = 4.833$, $P < 0.01$, $\eta^2 = 0.287$. The effect size for this statistically significant difference is large ($\eta^2 > 0.138$) (Cohen, 1988). Scheffe's post hoc procedures reveal that differences were

present in frequency of task changes between the late-mid and start phases ($p < 0.05$) and the late-mid and end phases ($p < 0.01$). These results support the earlier analysis, and show that in general, core developers were most active in the middle phases of their project (refer to Table 20 for details), and particularly during the late-mid project phase. Additionally, overall, while core developers made fewer task (WI) changes during the early project phases, these members were least active towards project completion.

Table 20. Percentage of overall task (WI) changes made by core developers over the duration of their Project

Team ID	Percentage of task changes			
	start	early-mid	late-mid	end
P1	19.4	26.2	25.2	29.1
P2	19.0	33.3	28.2	19.5
P3	35.3	14.7	32.4	17.6
P4	20.8	26.9	25.6	26.6
P5	28.9	19.1	37.5	14.5
P6	27.2	26.5	25.7	20.5
P7	22.1	32.6	23.2	22.1
P8	11.6	20.5	47.0	20.9
P9	20.8	26.0	32.5	20.8
P10	19.5	27.4	30.3	22.8
mean	22.5	25.3	30.8	21.4

Similar to the findings noted for core developers' contribution of knowledge sharing behaviours over their project, the results presented in this section reveal that these members did not contribute to task performance evenly over their project. Findings above show that core developers contributed the most changes to their teams' software tasks in the early-mid and late-mid project phases. Their contribution to task performance during the latter of the two phases was particularly pronounced, and the effect size associated with the difference observed in task performance is of major practical importance (Kampenes et al., 2007). On face value, these results support those revealed for the higher level of knowledge sharing behaviours that were contributed by core developers in the middle phases of their project (refer to Section 4.3.2). These results also coincide with use of delegation, insightful, certainty and positive emotion language by core developers as revealed in Section 4.3.1.

As noted earlier, given core developers' active role in their teams' behavioural climate, knowledge sharing and task performance, and their likely impact on their teams' overall performance, unearthing the way these members' contributions of attitudes and knowledge sharing behaviours are linked to their involvement in their teams' task

performance could provide a wealth of contingencies and recommendations related to team composition, software human resource planning and overall project governance. This issue is addressed next; first, core developers' actual contributions to task performance are correlated with their attitudes (Section 4.3.4). Subsequently, core developers' contribution of knowledge sharing behaviours is correlated with their task performance in Section 4.3.5.

4.3.4 Attitudes and Task Performance (RQ12)

This section presents correlation results between core developers' attitudes and their task performance. As observed above in Section 4.3.1, core developers expressed some key attitudes during specific periods of their project. Additionally, these members' task performance varied over their project (refer to Section 4.3.4). It is anticipated that core developers' involvement in software development activities may be associated with their expression of attitudes. For instance, it is likely that when core developers are most insightful and positive, these members may be more productive, and hence, useful to their teams. On the other hand, the opposite may be demonstrated when core developers are unhappy (Denning, 2012). An understanding of when these undesirable periods are most likely to exist would inform strategies aimed at mitigating the likely negative effects of this phenomenon on overall team performance. Accordingly, the results in this section are aimed at satisfying this objective in answering RQ12.

Before correlation procedures are conducted, scatter plots are examined for each of the thirteen linguistic dimensions and the task change data. In some cases, bivariate linear relationships between the two variables are observed, while in others this relationship is not apparent. Regarding the distributions, it is noted in Section 4.3.1 that ten of the 13 linguistic dimensions were within the range of normality, while three dimensions (collective (we), leisure and positive emotion (posemo) – refer to Table 5 for details) violated the normality assumption. Section 4.3.3 also shows that core developers' distribution of task changes were within the range of normality. These findings overall justify the use of correlation analyses, and more specifically, parametric correlation procedures (Pearson's product-moment correlation coefficient) for the normally distributed data and non-parametric correlation procedures (Spearman's rank order correlation coefficient – Spearman's rho) for the data that violated the normality assumption. These tests are conducted, and the results are now presented.

In Section 4.3.1 it is noted that core developers used more collective (e.g., we, our, us) language towards project completion, but used a similarly higher proportion of individualistic (e.g., I, me, my) language during the last project phase. While the correlation results did not reveal any relationship between individualistic language and the number of task changes, evidence of a small negative correlation between core developers' use of collective language and the number of changes they made is observed ($r = -0.211$, $n = 40$, $P = 0.191$); however, this relationship is not statistically significant ($p > 0.05$).

Similarly, although core developers were most active during the middle stages of their project, they also made the greatest use of reliance and delegation (e.g., you, your, you're) language during this time. This pattern was also seen for insightful (e.g., think, believe, consider) and discrepancy (e.g., should, would, could) language in Section 4.3.1. While there is no evidence of a relationship between the use of reliance language and the number of task changes, correlation results uncover a small positive relationship between insightful language use by core developers and the number of changes they made ($r = 0.129$, $n = 40$, $P = 0.428$). In contrast, when core developers communicated with higher levels of discrepancy language they made fewer task changes ($r = -0.128$, $n = 40$, $P = 0.431$). However, as with the result for collective language, these results are not statistically significant.

Of all the other correlation results, only work related (e.g., feedback, goal, delegate) language use is found to be correlated with core developers' task performance. In fact, this dimension reveals the only statistically significant ($p < 0.05$) positive relationship of all the 13 linguistic dimensions. It is observed that when core developers expressed more work related language they made more changes ($r = 0.360$, $n = 40$, $P = 0.023$), and this finding is reflective of a medium (Cohen, 1988) statistically significant positive relationship between the use of work related language and task changes.

Finally, the highest use of negative emotion (e.g., afraid, hate, dislike) language was expressed by core developers towards project completion, and Table 20 shows that these members were least active during this period. However, no association is observed between the use of this type of language dimension and the number of task changes made by core developers.

Overall, while many of the relationships between core developers' attitudes and their task performance were not statistically significant, this could be a function of simply having too few data points – thus, these results are reported here as they warrant further investigation. A similar approach is adopted in the following section (Section 4.3.5) where core developers' knowledge sharing behaviours are correlated with their involvement in task performance.

4.3.5 Knowledge Sharing and Task Performance (RQ13)

In order to answer the final question (RQ13) the connection between the directed CA (Section 4.3.2) and task performance analysis (Section 4.3.3) is examined. In line with the procedures used in the previous section (Section 4.3.4), Pearson product-moment correlation tests are conducted to determine the relationships between the knowledge sharing behaviours of core developers and the task changes they made. This procedure is fitting as none of the knowledge dimensions or the task changes data exhibited violation of normality, linearity or homoscedasticity.

For the Pearson's product-moment correlation results, a medium, negative correlation is seen between the number of Type I Questions asked by core developers and the number of task changes they made, although this is not statistically significant ($r = -0.390$, $n = 12$, $P = 0.210$). In contrast, when core developers initiated more Type II Questions in their dialogues they made more task changes ($r = 0.215$, $n = 12$, $P = 0.501$); this result is also not statistically significant ($p > 0.05$). The correlation test to determine the relationship between the number of Answers provided by core developers and their involvement in task changes uncover a strong, statistically significant ($p < 0.05$) positive correlation ($r = 0.691$, $n = 12$, $P = 0.013$). Similarly, correlation results for the relationships between the contributions of Information, Discussion and Scaffolding by core developers and their involvement in task changes uncover strong, positive correlations for Information provision and Scaffolding, ($r = 0.791$, $n = 12$, $P = 0.002$) ($r = 0.532$, $n = 12$, $P = 0.075$), respectively; however result for Discussion was not statistically significant ($p > 0.05$).

A medium, positive (but non-significant) relationship existed between core developers' contributions to Discussions and their involvement in task changes ($r = 0.466$, $n = 12$, $P = 0.127$). Additionally, a strong, positive correlation is observed between the volume of Instructions given by core developers and the number of task changes they made, but this finding is also not significant ($r = 0.539$, $n = 12$, $P = 0.071$). The Pearson's product-

moment correlation test to determine the relationship between the number of Comments contributed by core developers and the task changes they made shows some evidence of a medium, positive correlation ($r = 0.308$, $n = 12$, $P = 0.331$). However, these results are also not statistically significant ($p > 0.05$).

Attitude and knowledge sharing behaviour results in Section 4.3.1 and Section 4.3.2 are triangulated through Pearson's product-moment correlation tests to determine the relationship between core developers' use of the cognitive linguistic dimensions (see "insight", "discrep", "tentat" and "certain" in Table 5) and their levels of contribution to Information, Discussion, Scaffolding and Comments (refer to Table 6). Correlation results show a strong, positive correlation between the incidence of insightful language use and core developers' contribution of Information. This result is statistically significant ($p < 0.05$), ($r = 0.708$, $n = 12$, $P = 0.010$). A similar but less conclusive finding is noted for insightful language and Scaffolding ($r = 0.518$, $n = 12$, $P = 0.085$). Results for insightful language use and Discussion and Comments are less strong, but these instances also return positive correlations ($r = 0.470$, $n = 12$, $P = 0.123$) ($r = 0.345$, $n = 12$, $P = 0.272$). There is no evidence of any relationships between the other cognitive dimensions ("discrep", "tentat" and "certain") and core developers' contributions of Information, Discussion, Scaffolding and Comments.

These results confirm that core developers' contribution of knowledge was tied to their involvement in their teams' task performance. Overall, while some of the correlation results presented in this section were not statistically significant ($p > 0.05$), the findings show that core developers provided their teams more answers when they made more task changes. Similar patterns of results were noted for discussions when core developers were actively involved in their teams' tasks. These results for core developers' involvement in knowledge sharing were triangulated with their use of certain terms, and particularly those that were insightful in nature. This evidence may indeed inform team strategies and software project governance. These issues are considered in Chapter 5. Prior to this, a brief summary of the results that were presented in this chapter (Section 4.1, Section 4.2 and Section 4.3) is provided in the next section (Section 4.4).

4.4 Chapter Summary

This chapter has reported the results that were aimed at answering the 13 research questions (RQ1 – RQ13) that are outlined in Chapter 2. Results presented in Section

4.1.1 were aimed at answering RQ1, these revealed that Jazz teams typically communicated more in the start and end phases of their project. These findings are somewhat divergent to those discovered previously and echoed that the strategy selected for development likely impacts team communication patterns. Aimed at answering RQ2, Section 4.1.2 revealed that for all Jazz teams, few individuals dominated project interaction and these individuals established their position very early in their project. These findings provide support for the sampling strategy that was used for selecting Jazz teams' artefacts, and endorse previous evidence; however, these results failed to establish why such a pattern existed. With a view to understanding these central individuals further, results in Section 4.1.3 were aimed at answering RQ3, RQ4 and RQ5. These results revealed that core communicators were indeed core developers and these members communicated on seven times as many software tasks as their wider teammates. However, in terms of connectivity, all Jazz practitioners were observed to be highly reachable. That said, results presented in Section 4.1.3 show that core developers occupied various roles, with those assigned to the formal programmer role having the highest number of individuals in the core developers group. Additionally, it was revealed that core developers were integral to their teams' task portfolio. While results in Section 4.1 were able to identify the aforementioned patterns, the real reasons for core developers' dominance and how these members establish such a critical position in their teams were not revealed. This understanding was forwarded as particularly necessary given that core developers are critical to their teams' performance.

Accordingly, results in Section 4.2 were aimed at providing a static view of core developers' attitudes and their enacted roles, and in the process answer RQ6 and RQ7. Results in Section 4.2.1 show that apart from core developers' active involvement in task performance these members were integral for maintaining their teams' work and achievement focus. Contextual analysis findings in Section 4.2.2 also uncovered that core developers contributed most of their teams' knowledge, and these members occupied both task-based and social roles.

While these results established the true roles of core developers in their teams, this evidence provided a single snapshot view of these members' performance. It is held that temporal analysis would unearth specifically during which project phase(s) core developers are most productive and the specific events that influence core developers' actions throughout their project. Results from these inquiries were provided in Section

4.3 and answered RQ8 – RQ13. In the process of answering RQ8, in Section 4.3.1 it was discovered that core developers expressed relatively consistent attitudes over their project, but these members were most individualistic and negative towards project completion. These results coincided with those aimed at answering RQ9 and RQ10 in Section 4.3.2 for lower levels of knowledge sharing by core developers in the end phase of their project. Section 4.3.2 also discovered that core developers were most involved in their teams' knowledge sharing during the middle phases of their project. Results presented in Section 4.3.3 endorse this observation, where it was also shown that core developers made most task changes during the middle stages of their project. These results answered RQ11 and supported the assessment that core developers were most useful to their teams during the middle of their project. However, formal correlation testing was undertaken and reported in Section 4.3.4 and Section 4.3.5 to confirm this assessment and in the process answer RQ12 and RQ13. Results in these two sections (Section 4.3.4 and Section 4.3.5) confirmed that core developers' work-related expressions were driven by their active involvement in task performance, and their provision of answers and ideas were also directly related to their task performance. The approaches used to study core developers' attitudes and knowledge sharing were also triangulated towards the end of Section 4.3.5. The following chapter (Chapter 5) further discusses these results in relation to previous theories.

Chapter 5. Discussion

This chapter discusses the results reported in Chapter 4. First, Section 5.1 considers the preliminary findings in Section 4.1. This section discusses the collaboration patterns of successful globally distributed agile teams, focusing on the changes in communication patterns, the equity in practitioners' contribution, the importance of active communicators and their involvement in task performance and the way formal role assignment impacts practitioners' involvement in project interaction and task performance. The second section (Section 5.2) then discusses the results outlined in Section 4.2. These discussions centre largely on core developers' true role in their project, and particularly in relation to the behaviours and attitudes these members express and the roles they enact in their teams. The findings for these members' attitudes and enacted roles are compared to those of their lesser active counterparts. Third, Section 5.3 provides discussions of the findings in Section 4.3. These discussions consider core developers' attitudes, knowledge sharing and task performance from a longitudinal perspective, including how these members become hubs in their teams and the way their attitudes and knowledge sharing behaviours are linked to their task performance. Finally, the fourth section (Section 5.4) summarises the discussions that are provided throughout this chapter and provides the study's consolidated model.

5.1 Collaboration patterns (Phase 1)

Results in Section 4.1.1 to Section 4.1.3 are largely confirmatory of prior research outcomes. However, aspects of the findings discussed in these sections also extend those reported previously in the OSS body of work. Additionally, some new patterns were noted for IBM Rational Jazz teams that were not observed previously. This section examines these findings and discusses the preliminary quantitative results that were reported in Section 4.1. These discussions are aimed at understanding collaboration patterns of successful globally distributed agile software teams and forming the basis for other subsequent discussions, by answering the five preliminary research questions, RQ1 – RQ5. First, Section 5.1.1 discusses the ways in which teams' communicate over their project and addresses RQ1 (Do communication patterns change as the software project progresses?). Discussions are then provided in Section 5.1.2 that relate to the SNA results reported in Section 4.1.2. These discussions are presented largely from a project perspective and answer RQ2 (Is there equity in practitioners' contributions to their project?). RQ3 (Are active communicators more important to their teams'

collaboration?) is then discussed in Section 5.1.3. These preliminary discussions form the basis for further debates around core developers. RQ4 (How are active communicators involved in task performance?) is discussed in Section 5.1.4, where the results that were presented in Section 4.1.3 for active communicators' involvement in task changes are evaluated. RQ5 (Are practitioners' formal role assignments related to their involvement in project interactions and task performance?) is then discussed in Section 5.1.5. Finally, this section closes with a summary of the discussions presented throughout this section; this is presented in Section 5.1.6.

5.1.1 Communication patterns (RQ1)

Given the unique characteristics of globally distributed software development, and its growing use in industry, it is pertinent to study the way global teams communicate over their projects (Datta et al., 2011). Agile methods stress the need for incremental and iterative development. Such an approach is likely to have an effect on requirements solicitation and management. In particular, given that communication is the conduit through which software requirements are requested and clarified during development (and for collaboration during actual coding and testing activities), an iterative approach with a focus on extensive interaction in global software development (as is used in most agile development contexts) could provide added strain on team communication (Prikladnicki et al., 2003; Tiwana, 2004). In establishing how communication occurs in such a setting, insights could be gained into how globally distributed agile teams maintain the balance of being nimble while also delivering representative features and sustaining adequate plans for their global teams' awareness. In fact, beyond answers for how teams remain productive while being agile in a global development context, changes in the way project decisions are made (whether related to design, coding, or testing) could be revealed through the study of team communication (Abreu & Premraj, 2009; Bachmann & Bernstein, 2009). This approach to studying the project ecosystem as the project progresses has been taken in other development contexts (Capiluppi et al., 2007; Datta et al., 2011; Sharma & Kaulgud, 2011). In this study, this preliminary exploration is aimed at first understanding how the teams that were selected contributed messages over their project. In answering the research question (RQ1) in this section, this aspect of the work aims to provide preliminary understandings of distributed agile teams' dynamics, to replicate works that focused on other teams using different development approaches in other contexts (Robles et al., 2009; Rowley & Lange, 2007; Ruhnow, 2007) and to set the tone for the enquiries that follow.

The results reported in Section 4.1.1 are slightly divergent to those that were noted previously for the CSS teams studied by Datta et al. (2011). When studying Scrum teams Datta et al. (2011) found increased levels of collaboration as the project progressed, with communication reducing towards project completion. Such findings are understandable given the need to solicit requirements, develop these requirements, test these requirements and release them iteratively in a Scrum context (Pressman, 2009). Thus, over time, communications in such a development environment are likely to stabilise as this process is repeated, perhaps with reduced functionality as the project progresses these members need to communicate less, an assessment supported by Datta et al. (2011). Cataldo et al. (2006) study of a large distributed CSS project also found team communication increased as the project progressed. However, in contrast to Datta et al. (2011), these authors found team communication to increase in the final phase of the project, actually being at the highest towards project completion.

The results in Section 4.1.1 show that IBM Rational Jazz practitioners communicated the most in the first and last phases of their project. This finding would not be particularly surprising in a plan-oriented development setting, given the need to establish overall project goals and work assignments at the beginning of a project, to stabilise feature developments in the middle project phases, and then to intensively assess the overall project at its closure to ensure that the features developed match those requested (Abreu & Premraj, 2009). However, given that Jazz teams were using an agile-based approach (the Eclipse Way), these findings are somewhat revealing. The Eclipse Way methodology outlines iteration cycles that are six weeks in duration, where iterations comprise planning, development and stabilizing phases, after which builds are executed, conforming to key agile principles (“iterative, reflect, adapt, incremental, feedback”).

As noted in Section 4.1.1, however, Jazz teams engaged the most (perhaps in planning activities) around project initiation and in the early project phase. This is perhaps a good sign for distributed global software developments. The results revealed in Section 4.1.1 suggest that these teams expended more effort to adequately capture the overall project focus, before then stabilising iterations in the middle phases. Effort then increased at project completion as these teams validated the overall outcomes against the initial focus in the early project phase.

One caution about the results in Section 4.1.1 relates to the way artefacts were partitioned in order to be normalised. Datta et al. (2011) studied 10 different iterations in deriving their findings, while Cataldo et al. (2006) studied four releases. On the other hand, the project artefacts belonging to the ten teams that were examined in this work were associated with various numbers of iterations (e.g., P3 tasks were completed in two iterations, whereas P5 tasks were executed in 17 iterations) (refer to Chapter 4). To address this issue, these teams' artefacts were normalised by dividing each project's tasks and artefacts into four quarters (start, early-mid, late-mid, and end). Thus, the pattern of results could have been affected by this process. That said, these results are triangulated with other forms of quantitative and contextual analyses in subsequent sections. Note also that Cataldo et al. (2006) observed the similar elevated level of communication towards project completion.

Datta et al. (2011) warned that teams with communication patterns like those noted for the Jazz teams studied in this work may not be a good sign for project managers, as this may be an indicator of unbalanced project overhead. However, this form of communication pattern may be appropriate in a globally distributed agile software development context (Prikladnicki et al., 2003; Tiwana, 2004), where there is likely to be a need for balancing agility with some level of upfront planning (Sharma & Kaulgud, 2011). In fact, Yu et al. (2011) observed a similar pattern of communication in the OSS GNOME GTK+ distributed project for IRC meeting messages, where it was noted that contributors communicated significantly more in the early years (2004 and 2005) of the project, with communication via this means reducing in the middle years (2006 and 2007), before increasing in the latter two project years (2008 and 2009). The opposite pattern was noted for team communication when studying email messages, however (Yu et al., 2011). Yu et al. (2011)'s findings were also replicated by Shihab et al. (2009) work, who also found that contributors communicated much more at project start and completion.

In comparing the findings revealed from this preliminary exploration of Jazz teams' communications to those related to other teams, it is contended that different communication strategies are – and should be – adopted by different teams given the specific approach used for developing software. These discussions were derived from an aggregation of teams' messages, as against the internal structure of Jazz teams' communications. Such an internal assessment would provide further understanding for the ways in which these teams communicate. In incrementing the preliminary analysis,

these discussions are provided in the next subsection (Section 5.1.2), where the evidence discovered for the internal interaction patterns of Jazz teams is discussed.

5.1.2 Equity in contribution (RQ2)

Results in Section 4.1.2 show that, regardless of the task type, team size or time taken to complete software tasks, just a few team members dominated project interaction, and these dominant individuals emerged very early in the project lifecycle. Some of these members also worked across project teams, and occupied similarly central roles. In terms of the less active members, their engagements were also consistent, and these members communicated densely on some specific tasks. These tasks were possibly directly under these members' control.

The findings in Section 4.1.2 are slightly divergent to those reported in the work of Nguyen, Wolf, et al. (2008) which found a larger cohort of Jazz developers (around half of the practitioners) to be involved in the core of their teams communication network. Nguyen, Wolf, et al. (2008) studied an earlier version of Jazz, and did not employ a fine-grained approach to their analysis as was done in this work. These differences may account for the divergence in patterns observed. In Nguyen, Wolf, et al. (2008) work, SNA was used to explore all the communication artefacts in the Jazz repository, and the k-cores were examined closely to study dense network segments. Nguyen, Wolf, et al. (2008) examined these segments to see if members in these sections of the network had higher network ties (communicated on more tasks). In contrast, this work mined Jazz from a team perspective, and clustered artefacts according to the natural partitioning of the Jazz repository (in terms of project areas; see Section 3.4.2 for details).

Additionally, the k-core measures of the individual teams were not actually examined in this study as the goal of this work was to go beyond the assessment of message frequency to employ multiple techniques to the study of teams' communication.

In Section 4.1.2 it was observed that once the central communicators' positions were established, their roles in their teams' communication networks remained consistent over the course of the software project. These observations seems to denote that oftentimes the holders of knowledge in a software project, once recognised, are regarded as such for the duration of their involvement with the team; knowledge sharing here being considered as the quantity of information flow (Quigley et al., 2007). Such a communication norm, once deemed necessary, is thought to be sustained by highly motivated team members (Chang et al., 2013). Perhaps the less central team members

acknowledged these members as knowledge hubs once their capabilities were established very early in the project. This was also observed by Shihab et al. (2010). In fact, this pattern has been previously recognised by others studying developer messages and change logs. For instance, Shihab et al. (2009) found that 20% of the core developers communicated 80% of the team's messages in the Evolution and GTK+ mailing lists. Others have also found this structure to exist in other OSS settings (Crowston et al., 2006; Gacek & Arief, 2004; Mockus et al., 2002).

This centralised pattern is somewhat understandable for OSS environments, given that individuals contribute voluntarily to such projects for reasons often associated with personal interest and ideological commitment (Ljungberg, 2000; Markus, Manville, & Agres, 2000; Oreg & Nov, 2008), or even in order to gain skills and enhance their reputation (Markus et al., 2000; Oreg & Nov, 2008). Thus, while many contributors may *join* OSS projects, during stressful and challenging stages of these projects members may not sustain their initial interest, and are likely to leave the project or at least not actively contribute. This phenomenon has been shown to exist where, apart from the core group of developers, most other members were seen to generally contribute in a sporadic manner (Crowston et al., 2004). On the other hand, the core group of members in OSS settings whose motivation and rewards may be different to the less active members (Gacek & Arief, 2004) (e.g., sponsorships or the development of a commercial variant of the OSS), are likely to remain with these projects.

Previous research has also expressed caution regarding inferences and generalizations drawn from analyses of the extracted OSS repositories that are regularly used to study software practice issues, due to questions over reliability and validity (Aune et al., 2008). For instance, as noted previously, research evidence has reported poor data quality in some repositories OSS projects (Aune et al., 2008; Bird et al., 2006a). In their study of the Apache mailing list Bird et al. (2006a) found it difficult to uniquely identify developers' records due to the volume of email addresses and aliases individuals used. Issues may also be encountered when studying OSS repositories because anyone is able to post messages to these mailing lists, whether or not those individuals are contributing to the project (Bettenburg et al., 2007). In fact, evidence has shown that the majority of OSS mailing list members are not involved with the actual development (Bettenburg et al., 2007), but are instead regular users of the software who communicate their interest to the core developers and report bugs (Crowston et al., 2006), and so many contributors to the OSS mailing lists could not be accounted for in code changes (Bettenburg et al.,

2007). In IBM Rational Jazz, however, those communicating on features are actually part of the teams undertaking these software tasks.

Given these latter arguments, it would be understandable to observe centralised communication and administration for projects developed in an OSS environment. Core developers perhaps work in isolation to deliver features and fixes in response to user feedback in this setting. However, the opposite is thought to be necessary in a commercial organisation such as for IBM Rational Jazz (Robles et al., 2009), where teams operate under a solid project vision, utilising tested software processes, and their motivation is much more than personal satisfaction. In fact, developers' motivations in commercial projects are quite different to those in OSS development settings (there is a large literature base on software engineers' motivations (Sach et al., 2011; Sharp, Baddoo, Beecham, Hall, & Robinson, 2009)), and the rewards offered in this setting are immediate (e.g., financial remuneration). In these settings software practitioners may feel a sense of moral obligation in direct recognition of remunerations and rewards (Kankanhalli, Tan, & Wei, 2005). Additionally, commercial environments often revise work strategies to fit individual practitioners' coordination needs. In particular, communication strategies are likely to be aligned with the teams task portfolio in commercial organisations, such that when teams are working on large numbers of interdependent features where communication among the entire team requires careful management to ensure efficiency and team awareness, specific mechanisms are likely to be implemented in support of this phenomenon (e.g., communication plans and expertise awareness tools) (Grinter, Herbsleb, & Perry, 1999; Mockus et al., 2002). The very high level of network connectivity that was observed for all ten Jazz teams considered here endorse this viewpoint (refer to Section 4.1.2). Successful commercial software organizations are also likely to implement human resource management strategies to ensure intense screening of selected practitioners, and especially in relation to communication skills (Colomo-Palacios et al., 2010; Downey, 2009). Furthermore, task assignment is generally managed by the project manager in commercial software environments (Mockus et al., 2002, p. 344; Robles et al., 2009).

With these assessments in mind, of further relevance to this discussion thread is the observation that all of the ten teams studied in this work had a centralised communication structure (regardless of the number of tasks or team size). This pattern has been explained in terms of several prior principles. For instance, Shihab et al. (2010) used the Pareto principle to explain this pattern in GNOME OSS projects, where

they found the top 10% of communicators contributed 60% of the overall messages. The Pareto principle is that a minority of the cause influences the most effect (Shihab et al., 2010). This principle was also previously demonstrated by Boehm & Basili (2001) who discovered that 20% of the code accounted for 80% of the bugs in the software they studied. Another principle that has been previously associated with this pattern is that of small-world communication (Uzzi & Spiro, 2005). This principle is used to describe interaction structures where small groups of contributors share most of their communication among a core cluster of team members while working in a larger team. However, members in the core cluster may have connections to others in multiple clusters, which then connect with others in other clusters, making the overall communication network connected and cohesive (Uzzi & Spiro, 2005). Andre, Baldoquin, & Acuna (2011) study on role distribution also found that successful teams had few leaders who were consistently strong communicators. Thus, it would be reasonable to assume that the centralised pattern that was observed for these Jazz teams is linked to role distribution (or the leadership hierarchy). However, the results presented in Section 4.1.3 do not support this latter proposition, as in this section it was observed that many of the active communicators were not formal project leaders, and thus, were not necessarily responsible for project coordination.

The real question then becomes why there are disparities in communication for software development teams, and in particular, how important are these active communicators to their teams in terms of maintaining team connectivity? The next section (Section 5.1.3) considers the question of active communicators' importance. Section 5.1.4 then addresses active communicators' actual involvement in task performance.

5.1.3 Active communicators importance (RQ3)

This section discusses the importance of core developers to their global teams' collaboration. Given the dense communication patterns that are noted around central software practitioners (see Section 4.1.2 and Section 4.1.3), these members are held to be critical to their teams' shared understanding (Crowston et al., 2006; Mockus et al., 2002). This deduction is rational, as one would expect that the removal of such members from their teams would erode key links to their teams' less central and peripheral members. Such links are likely to be critical for maintaining shared perceptions and a friendly team climate, and promoting team optimism or urgency in the face of schedule pressures. Such a position has been established in other disciplines

(Ahuja et al., 2003; Guetzkow & Simon, 1955), and early works investigating the significance of centralised group members have stressed the importance of these actors to their team's performance (Bavelas, 1950; Leavitt, 1951), and particularly for their teams' information dissemination.

Thus, in making provision for these members' reduced availability, absence or sudden withdrawal from the team, project management may also promote team configurations that are likely to provide failsafe mechanisms. In fact, the threat imposed by the loss of key team players (and with them – the team's tacit knowledge) has been a recognised source of concern for agile teams given their reliance on team members' interaction as a substitute for extensive documentation (Boehm & Turner, 2003a, 2003b; De Souza, Anquetil, & De Oliveira, 2005; Nord & Tomayko, 2006). This threat is likely to be exacerbated for globally distributed agile software teams, where there are limited possibilities for spontaneous and informal communications (Serce et al., 2009). Studying teams' communication is likely to expose how this issue is addressed, and should also help to explain the importance of active communicators to their global teams (Abreu & Premraj, 2009; Bachmann & Bernstein, 2009).

During the analysis of Jazz teams artefacts in Section 4.1.3, active communicators were clustered into a group called core developers and their importance was compared to that of the other team members. Apart from the higher frequency with which core developers communicated, there was also vast disparity between the numbers of tasks core developers communicated on compared to the rest of their team members (refer to Section 4.1.3). The results show that core developers communicated in relation to seven times as many tasks as their teammates, on average. These findings show that, beyond the incidence of messages, core developers also maintained interest on many tasks. This high level of interest may not necessarily denote that core developers are important, however. For instance, it has been previously contended that these members may be involved in team task coordination and liaison (Cataldo & Herbsleb, 2008). Others have also stressed the need for studying the *actual* discussions and project documentation to understand the actions and attributes of real core developers (Robles et al., 2009). Accordingly, mechanisms for studying core developers' level of importance (in-degree and closeness) to their teams are described in Section 3.4.3.

Section 3.4.3 outlined that SNA in-degree measures the number of connections that point towards a vertex and closeness measures the shortest distance between nodes, so

that the lower the closeness measure for a given node the more important that node is to their group's communication (Wasserman & Faust, 1997). These measures were considered jointly in Section 4.1.3 in order to study core developers' importance to their team (Bird et al., 2006a; Datta et al., 2010). That said, given that these measures only reflect a form of structural importance, rather than intellectual or social importance, there is need to use other deeper contextual analysis techniques to validate these findings. This approach was used in this work, as further discussed in Section 5.2 and Section 5.3.

In analysing the results for core developers' in-degree measures it was observed that these members contributed over a fifth of their teams' measures. These findings endorse the density measures noted earlier. However, core developers' closeness measures, although lower than those of their teammates, were not significantly different. In fact, Section 4.1.3 outlined that even when the communications for core developers were removed from their teams' social networks, these networks still remained highly connected. From these findings it is surmised that while Jazz core developers were highly active in communication, and they perhaps occupied critical links in their teams' shared understanding processes, maintaining a friendly team climate and promoting team optimism and urgency when necessary, the social network measures did not find these members to be significantly more important than the less active members in their teams, in terms of maintaining the connectivity of the members at the network's periphery. Less active members were connected through their involvement on software tasks, whether or not core members were present. This finding has not been observed previously.

Notwithstanding the highly centralised nature of IBM Rational Jazz teams' communication, this is a positive observation for these teams, and this evidence may have implications for the configuration of globally distributed agile software development teams in a more general sense. In particular, given that these are high-performing and successful teams, this outcome is something that others could seek to replicate. Overall, the results revealed that although some Jazz members communicated on few tasks, all team members remained highly reachable and connected either through their engagement on other tasks or directly through their connections. Although it is unclear whether this is a deliberate strategy employed by Jazz teams, or whether this evidence reflects the process of self-organisation among high performing globally distributed agile software practitioners, this finding supports the synthesis above

regarding the ways in which commercial organisations are driven, and the motivations of developers in such settings (refer to Section 5.1.2). A strategy that promotes interconnected communication across software tasks may work as a cross training mechanism (Highsmith, 2000, 2004). Such a coupling strategy is also likely to result in a higher degree of knowledge sharing and reduced loss of tacit knowledge should core developers leave these teams (Boehm & Turner, 2003b; Williams & Kessler, 2003).

In summary, the findings revealed in Section 4.1.3 are surprising in several ways. First, and given prior research outcomes, it was anticipated that there would be a much larger spread of team members involved in team interactions. In particular, higher levels of modularity was expected for teams that were solving a larger cohort of tasks due to the heavy knowledge demands associated with managing larger projects (Mockus et al., 2002). That said, however, the difficulties associated with managing a large number of connections could also present a burden to these teams. Thus, a centralised structure is likely to reduce project communication overhead and costs related to coordinating a large number of team dependencies (Crowston et al., 2004). A strategy to make teams highly connected would then mitigate the effects of tacit knowledge loss should core developers leave their teams (Williams & Kessler, 2003). Thus, the centralised structure noted in Jazz networks may not necessarily introduce risks to these teams.

That said, although the SNA results did not establish that core developers were most important to their global teams' collaboration, it is still pertinent to establish how core developers are involved in task (WI) changes, as this would reveal further insights into global team dynamics. Such evidence would also provide additional understandings into these practitioners' roles during teamwork. This issue is considered in the next subsection (Section 5.1.4).

5.1.4 Active communicators task performance (RQ4)

Those that are formally assigned to coordination inclined software roles (e.g., team leaders and project managers) may generally be expected to communicate more than the average software programmer (Shihab et al., 2010). Accordingly, while these practitioners are likely to provide project awareness and guidance to their teams, they are consequently unlikely to be core developers on actual software tasks. The goal in this work is to understand the collaboration patterns of successful globally distributed agile software teams, and how and why core developers contribute to globally distributed agile software team dynamics. While the first object is easily realized by

studying overall teams' artefacts (Nguyen, Wolf, et al., 2008), the latter objective could only be achieved by studying actual core developers – those that communicate extensively and also demonstrate key involvement with their teams' actual development portfolio (Cataldo & Herbsleb, 2008; Crowston et al., 2006; Robles et al., 2009).

In fact, as noted in Section 5.1.3, while it was established that a few individuals tend to dominate project interaction, SNA results in Section 4.1.3 revealed that these core members were not significantly more important to their teams than the less active members (refer to Section 5.1.3 for further details). Accordingly, core developers' actual involvement in software tasks was examined in Section 4.1.3, and the results showed that core developers played a key role in their teams' task portfolio. This finding was also noted previously in both OSS and commercial settings (Cataldo & Herbsleb, 2008; Shihab et al., 2010). For instance, Mockus et al. (2002) observed that the top 15 developers contributed 80% of the code for new software functionality in the Apache httpd OSS project. In contrast, these members (i.e., those who were assigned to the top developers group) only reported 5% of the bugs on their project. Shihab et al. (2010) found a correlation between the number of messages practitioners communicated and the number of code changes they made, when studying the Evolution and Nautilus OSS projects. Bird et al. (2006a) found that practitioners' communications were strongly related to their involvement in source code changes, and a similar pattern was revealed in Cataldo & Herbsleb (2008)'s work.

These findings seem to suggest that task involvement influences the need to communicate; perhaps the communication pattern is influenced by the management of a large number of feature dependencies. Discussions around these dependencies are likely to be particularly pronounced at integration time. Thus, the more software features developers deliver, the more they are required to communicate (Cataldo & Herbsleb, 2008). The findings revealed in this work, and by those in other studies (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Shihab et al., 2010), are surprising given the overhead associated with 'owning' large numbers of features, particularly in a distributed development setting where temporal distance may affect team members' availability (Espinosa et al., 2006). Reduced availability could potentially hamper core developers' engagement possibilities (Carmel & Agarwal, 2001; Cataldo, et al., 2007; Herbsleb & Mockus, 2003a; Jalali & Wohlin, 2010), and ultimately, discussions around decisions that are made during feature development (although given the high number of

messages IBM Rational Jazz core developers sent, this may not be an issue for these particular developers).

In fact, while centralised feature management (in terms of reviews and approvals) may help with team productivity and lessening defect density, such an approach generally extends development time frames (Mockus et al., 2002). Of particular note is the way core developers were integrally involved in task modifications. Given this evidence, there is some contradiction of the view that the pattern around core developers may be as a consequence of a deliberate strategy that is implemented at IBM Rational Jazz to maintain productivity and quality (this speculation was forwarded during discussions in Section 5.1.2).

On the contrary, IBM Jazz core developers seem to evolve naturally into their central role. Perhaps these members are of particular demand because of their natural characteristics. This assessment is further verified by an investigation of their formal roles through contextual analysis techniques (see Section 4.1.3). These discussions are provided in the next section (Section 5.1.5).

5.1.5 Active communicators formal roles (RQ5)

Discussions presented thus far highlight that IBM Rational Jazz developers spend the most time communicating in the start and end phases of their project, a few developers dominate project interaction, core developers are not significantly more important than their less active counterparts, and core developers play an integral role in their teams' development portfolio. These findings were discussed in relation to relevant theory in Sections 5.1.1, Section 5.1.2, Section 5.1.3 and Section 5.1.4 above. The final aspect in this preliminary phase of the work relates to the assessment of core developers' formal roles.

Results presented in Section 4.1.3 revealed that formal role assignment did not limit IBM Rational Jazz core developers' performance in communication networks or on software development tasks. Those leading the interaction networks occupied various roles – including programmers, team leaders and project managers. Given the high number of task (feature) changes undertaken by core developers, it was also posited that these members were likely to communicate actively due to task dependencies. In fact, results in Section 4.1.3 show that in a slight majority of the cases core developers (both in communication networks and involvement in software tasks) were programmers.

These findings are interesting, given that IBM Rational Jazz teams are each led by a formal team leader. Thus, it was expected that those assigned to leadership roles (team leaders and project managers) would at least dominate project communication networks given their need to coordinate and manage multiple project dependencies. However, the evidence provided in Section 4.1.3 is clearly to the contrary.

Agile practitioners have previously reported that team members adopted various roles over different project phases in order that their projects should succeed (Hoda, Noble, & Marshall, 2010b). Datta et al. (2010) SNA study also found that some team members' actual involvement in bug fixes exceeded what was expected given their formal roles. These findings support the proposition made above in Section 5.1.4, that the core developers studied in this work may be driven by some specific intrinsic characteristics and/or motivations. Evidence in Section 4.1.3 shows that core developers were not restricted by their formally assigned responsibilities; rather, these members seemed to perform given the teams' demand. This assessment is particularly fitting for those core developers who formally occupied the programmer role.

These may not be default behaviours, however. While core developers may feel a sense of obligation to their teams (Constant, Sproull, & Kiesler, 1996), a facilitating organisation and work structure may be a prerequisite for encouraging high performers to work across roles as the need arises. Given the evidence revealed in this work, it is posited that IBM Rational is one such organisation that encourages team members' performance based on their natural abilities, and that promotes non-hierarchical and informal work structures. Such configurations have long been shown to encourage tacit knowledge sharing and cross-fertilization among team members, and allow team members to adapt and execute their tasks based on work demands (Powell, 1990). These environments are well suited for globally distributed agile software development teams, and should be encouraged if such teams are to succeed.

In comparing the outputs of this role examination to previous literature, it is noted that previous studies have speculated that programmers require fewer communication-related abilities (Acuna et al., 2006; Andre et al., 2011). However, the evidence reported here is divergent to these views. Results in Section 4.1.3 confirmed that *all* software practitioners may actively participate in communication and coordination networks if/when the project environment is supportive. In fact, closer examination of the roles for the 15 core developers revealed that role assignment was not a barrier for any of the

teams, and in a number of the ten teams that were studied both core developers were formally assigned the programmer (contributor) role. These similar findings for Jazz core developers across all ten project teams – regardless of the nature of the tasks, or number of team members – support the position that the IBM Rational Jazz work structure had a positive influence on the way core developers were able to interact and share knowledge (Giddens, 1979; Orlikowski, 1992).

Additionally, the evidence observed in Section 4.1.3 regarding the way those assigned to different formal roles became communication hubs in their teams supports the view that IBM Rational Jazz teams were encouraged to work across roles and self-organise. Self-organising theories have noted that a prerequisite for successful self-organising practitioners is the ability to work across multiple roles (example: roles that contribute ideas, facilitate coordination and communication and remove obstacles) (Hoda et al., 2010b). Thus, although practitioners are assigned formal roles in IBM Rational Jazz teams, during project execution it is believed that core developers enact other non-formal roles, perhaps so that their project's communication and coordination requirements are met, and their project succeeds. This finding has implications for globally distributed agile software development teams, and particularly, for instances where such core developers may be unwilling to accept project leadership and champion responsibilities.

In summary, although the IBM Rational Jazz teams studied here operated in a centralised structure, it was common to see many programmers occupying vital positions in their team's network. While the literature advocates for non-centralised and non-hierarchical structures especially in distributed software development contexts (Crowston & Howison, 2006), these Jazz teams are likely to have been insulated from negative issues related to over-centralisation due to the way IBM Rational Jazz team members are involved with software tasks – these members are highly connected whether or not core developers are a part of their communication networks (refer to Section 5.1.3). In fact, the burden associated with information processing for groups with higher numbers of inter-connections may in some way be lessened in the Jazz teams' context (Hinds & McGrath, 2006). Additionally, although this study does not examine the actual skills of Jazz developers, this may also mitigate the effects of centralised communication structure. It is posited that Jazz teams possess premium skills given their project portfolio, the project range available at <http://www.jazz.net>, the

volume of customers using these products and the sentiments expressed by these users (refer to Section 3.4.1).

5.1.6 Summary

Figure 29 depicts the main findings that were discussed in this section. In this preliminary quantitative analysis of IBM Rational Jazz globally distributed agile software teams' communication, it was revealed that these teams communicated the most at project start-up and towards project completion (refer to Section 5.1.1), only a small number of individuals dominated project communication networks and early communication patterns were maintained throughout the software project (refer to Section 5.1.2). Additionally, findings in this work revealed that communication networks with low density may not necessarily mean that contributors are not reachable, as all members of the ten IBM Rational Jazz teams examined here were highly connected, whether or not they belonged to the core developers' group (refer to Section 5.1.3). Given this evidence, it is contended that the centralised communication pattern noted for Jazz teams may not necessarily represent a risk related to the loss of the project's tacit knowledge should core developers leave these teams (refer to Figure 29). Additionally, it was contended that this highly connected pattern for Jazz communication networks may be linked directly to deliberately adopted organisation processes to deal with maintaining knowledge redundancy, in order to address risk related to tacit knowledge loss.

The analysis in this section also demonstrated that those leading interaction networks were heavily involved in performing software tasks (refer to Section 5.1.4).

Additionally, it was found that core developers occupied various formal roles (refer to Section 5.1.5). These findings were consistent for all teams, an indicator that the IBM Rational Jazz organisation's flexible work structure may have positively influenced the way teams interacted and core developers' willingness to adopt project champion roles even when they were not formal leaders. This finding was particularly revealing for those core developers that occupied the programmer role.

While some of the findings in this initial analysis are confirmatory of prior outcomes, others were unique and point to the need for further in-depth qualitative exploration of these artefacts. In particular, the quantitative findings that were discussed in this section did not reveal the *semantics* of core developers' behaviours and attitudes. Although previous work has linked practitioners' performance to experience and cognitive ability

(Curtis, Krasner, & Iscoe, 1988), the nature of core developers' characteristics may not be entirely explained through quantitative means, and more contextual analysis may reveal relevant attributes of these practitioners' behaviours and the actual roles they enact during teamwork (Cataldo et al., 2006). For instance, Cataldo et al. (2006) previous work did not find significant differences between core developers and their less active counterparts when considering their programming experience, domain experience, education or tenure in the company. Similarly, productivity measures drawing on mean lines of code for features did not reveal any differences between core developers and the other members in their team.

To this end, it is believed that the use of other analysis techniques commonly employed in organisation psychology and the social science domains could help to reveal further the nature of (and reason for) core developers' attitudes, and the roles these members enact during globally distributed agile software team dynamics. These understandings would further illuminate the nature of globally distributed software development team dynamics. This approach was therefore used in this work, the results of which are discussed in the following section (Section 5.2).

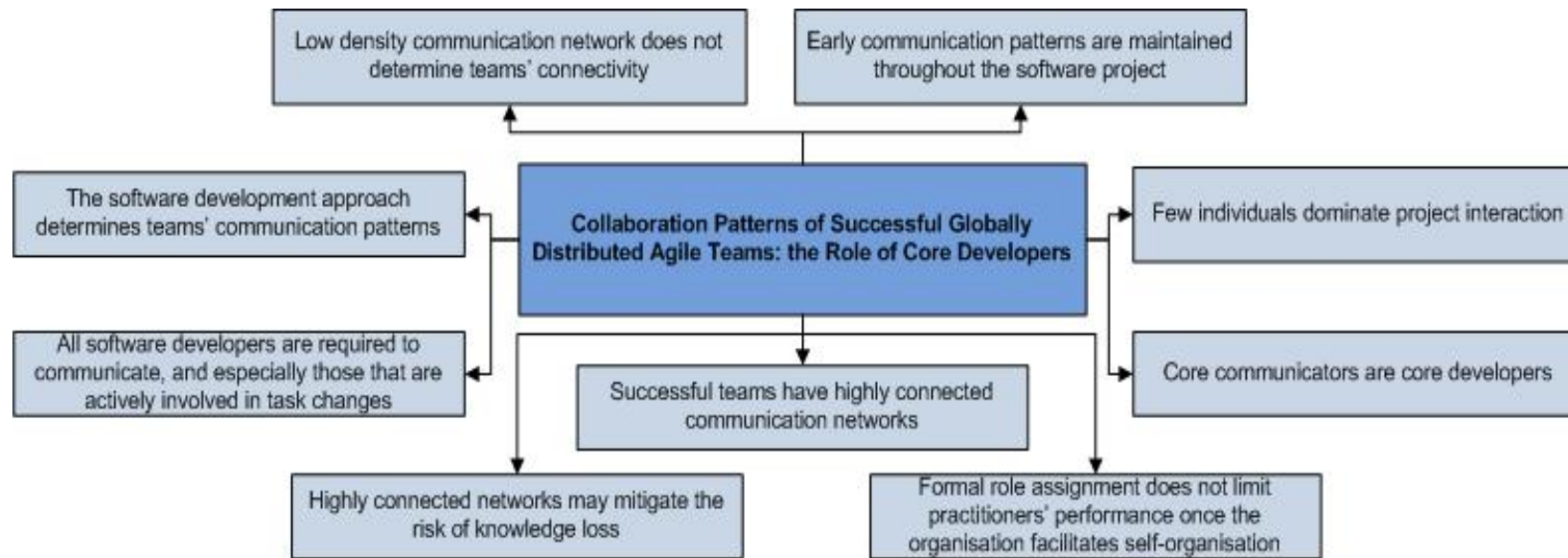


Figure 29. Collaboration patterns of successful globally distributed agile software teams

5.2 The true role of core developers (Phase 2)

The results presented in Section 4.2.1 and Section 4.2.2 are largely exploratory, extending those of previous studies as well as the evidence proffered in Section 4.1. These findings thus provide understandings of the true role of core developers and explain the reason for the centralised patterns noted for software teams' communication networks. Through such understandings, these findings aimed to establish the reason for core developers' distinct presence in their teams, and to provide explanations for the nature (and peculiarities) of successful distributed agile software team dynamics. More specifically, it is anticipated that knowledge of how the most active agile practitioners (and others) contribute their social and intellectual capital would help project leaders to identify exceptional software practitioners, and inform the process of assembling high performing and cohesive teams. Such findings could also inform the use of specific organizational arrangements and team configurations in support of high performers. Furthermore, the output of these explorations may lead to new requirements for collaboration and process support tools.

This section discusses these findings by answering RQ6 and RQ7. First, Section 5.2.1 discusses the ways in which core developers' attitudes differ to those of their less active counterparts, so answering RQ6 (Do core developers' behaviours and attitudes differ from those of other software practitioners?). Results for RQ7 (What are the core developers' enacted roles in their teams, and how are these roles occupied?) are then discussed in Section 5.2.2. Finally, this section closes with a brief summary (presented in Section 5.2.3) of the discussions presented throughout this section.

5.2.1 Differences in attitudes (RQ6)

Evidence discussed in the preceding section confirms that a small number of IBM Rational Jazz team members (the core developers) contributed the most to their project's knowledge-base and these members were also highly involved with software task changes. This evidence has been observed previously for other software teams (Bird et al., 2006a; Shihab et al., 2010). Given this pattern, along with evidence of core members' influence on their teams' overall performance (see Leavitt (1951) for discussions), it is imperative to understand core developers' communications beyond frequency-based assessments alone (as was done in Section 5.1) (Di Penta, 2012). More in-depth evaluations could potentially answer questions related to the reasons for these members' extraordinary presence, and provide understanding of the actual roles (both

formal and informal) that core developers occupy in their teams. This is particularly necessary given that measures related to education, experience and cognitive ability (Cataldo et al., 2006; Curtis et al., 1988) have not explained the differences noted in communication and software task involvement patterns between core developers and their lesser active counterparts. This work examines these differences from a behavioural perspective. In this section, discussions of the linguistic analysis results (project snapshots/static perspectives – refer to Section 4.2.1) are presented.

The linguistic analysis findings in Section 4.2.1 show that, when compared to the less active software practitioners, core developers were less self-focused (or individualistic) and, although these individuals were most actively involved in task changes (refer to Section 5.1.4), they delegated more. Members that are individualistic have been shown to have a negative impact on team climate (Benne & Sheats, 1948); these individuals are seen to be driven by their own personal goals as against those of the team (Stone & Pennebaker, 2002). Thus, the evidence that core developers exhibited lower levels of self-focus attitudes is a good sign for shared team norms (Chang et al., 2013). This is particularly noteworthy given that, as discussed in Section 5.1.2, core developers communicated in regard to a substantial number of their teams' tasks. Had these members exhibited high levels of individualistic attitudes, this would potentially impact their teams negatively, and particularly given that these core members occupied the centre of their teams' communication, and so, behaviour climate, team culture and trust (Dullemond et al., 2009; Lee & Yong, 2010) – all of which have been held to be necessary and critical to teams' performance in globally distributed settings. Evidence has indeed shown that team norms are cultivated (Chang et al., 2013; Denning, 2012). Given that software development is a shared activity, individualistic team norms could have a negative impact on team performance, and particularly in a globally distributed agile software development context where individuals are already affected by distance and have few opportunities to engage in face-to-face communication (Chang & Ehrlich, 2007; Espinosa et al., 2006) – both of which are known to stimulate trust (Al-Ani et al., 2011; Krebs et al., 2006; Ziguers, 2003).

Findings revealing the higher level of delegation among core developers are surprising given that core developers comprised only 15 of the teams' 146 practitioners. However, these findings are supported by Shihab et al. (2009) who also found that the Evolution and GTK+ OSS projects' top developers referred to others by their actual names (directly addressing contributors), an observation that was linked to their status in the

team. As noted earlier, however, team dynamics and processes in OSS environments are somewhat different to those in commercial organisations (refer to Section 5.1.2). In fact, among IBM Rational Jazz's less active practitioners there were many formal project managers and team leaders (see Table 2 for details), whereas core developers comprised just eight programmers, five team leaders and two project managers.

This evidence of the much higher level of delegation among core developers suggests that these practitioners indeed operated as informal leaders (an assessment that was incited in Section 5.1.5), whether or not they were assigned to formal leadership roles (Licorish & MacDonell, 2013c). This assessment converges with those in Section 5.1.5 above, where it was postulated that IBM Rational Jazz core developers operated freely across roles. Additionally, this evidence also supports the notion that IBM Rational Jazz organisation promoted organic work structures, and this in turn encouraged practitioners' performance in light of their teams' demands, policies that may be linked directly to Jazz teams' success (refer to Section 5.1.5). These findings have implications for globally distributed agile software teams, and particularly those environments that employ rigid project management approaches (Coram & Bohner, 2005). Such a tactic may be detrimental in globally distributed software environments. Evidence in this work also suggests that formal project leaders may need to regularly compromise, perhaps encouraging, and releasing control to, informal leaders as the need arises (Abrahamsson et al., 2003). These principles are generally embraced by agile proponents (Koch, 2005).

Linguistic analysis results for work- and achievement-processes add further support for these deductions. Results in Section 4.2.1 reveal that core developers were highly work- and achievement-focused, and these individuals communicated little about leisure. Rigby & Hassan (2007) also found top Apache OSS developers to be less social than the other members. Previous work has noted that individuals that are highly ambitious are generally outcome-oriented (Denning, 2012). These individuals are often committed to succeeding regardless of the circumstances (Chang et al., 2013). Well-established role theories have also shown that task-driven individuals are keen on task performance and drive their teams towards achieving project targets (Belbin, 2002; Benne & Sheats, 1948). Thus, the finding that core developers were highly task-focused is fitting for their teams, and these results are positive for the IBM Rational Jazz organisation. In particular, given core developers' central position in their team communication networks (noted earlier – refer to Section 4.1.2 and Section 4.1.3), these practitioners

were uniquely positioned to promote team urgency during times of schedule pressures and when the less active members demonstrated reduced task focus. Additionally, as noted above, core developers' ubiquitous qualities would mean that their achievement-driven behaviours are likely to easily propagate to their teams, resulting in achievement-driven team norms (Denning, 2012). Such team norms would be ideal for globally distributed agile software teams' performance (Abrahamsson et al., 2003).

These assessments are of particular relevance given the results that were discovered in Section 4.1.3 through the use of quantitative SNA techniques. These results were discussed in Section 5.1.3, where it was asserted that although core developers contributed significantly more communication, these members were not significantly more important to their teams than the less active developers (where importance was evaluated through the use of in-degree and closeness measures – refer to Section 3.4.3.1 for details). These linguistic analysis results contradict these initial findings (in Section 4.1.3), and show that core developers exhibited important team behaviours (discussed earlier). These findings endorse the view that, when used on their own, quantitative and frequency-based analysis techniques illuminate only a partial view of team dynamics (Di Penta, 2012; Easterbrook et al., 2008; Glass et al., 2002; Robles et al., 2009; Vessey et al., 2002), and complementing these approaches with contextual analysis techniques would further reveal the intricacies and complexities of human behaviours and software teams' dynamics (Vessey et al., 2002).

Results in Section 4.2.1 show that core developers were less positive and less social than the other team members. Perhaps IBM Rational Jazz's core developers were too involved with their teams' development agenda to be social. Findings for these members' involvement in task (WI) changes and communication (refer to Section 4.1.2 and Section 4.1.3) support such a conjecture. However, there still remain questions around what actually drives these members' motivation. While this work was not able to examine the experience and education of core developers, as noted above, these variables did not account for differences noted between core developers and their less active counterparts in other settings (Cataldo et al., 2006; Curtis et al., 1988). Previous work has explained that strong team commitment is linked to sentimental attachments to team goals (Allen & Meyer, 1990; Morgan & Hunt, 1994) and the willingness to maintain team commitment through team tasks (Mowday, Steers, & Porter, 1979). Highly committed team members are also said to feel a sense of team duty and have a strong desire for team success (Kline & Peters, 1991). Such team members exhibit

extreme willingness (Oh, Gallivan, & Kim, 2006). Dedicated team members are also willing to adjust themselves in providing relevant knowledge for their team (Lee & Xia, 2005). The concept of the super-individual entity in team work has also been explained previously (Walsh, 1995), and such individuals are emphasised as necessary for agile software development settings (Abrahamsson et al., 2003). These viewpoints may explain IBM Rational Jazz core developers' performance in their teams. While there is little doubt that Jazz core developers were highly cognitive, these members were very committed to their teams' tasks. Such a commitment is likely to be driven by more intrinsic than extrinsic motivations (e.g., from personal satisfaction and enjoyment rather than from interest in organisational rewards) (Chang et al., 2013).

The linguistic analysis results (refer to Section 4.2.1) revealed that the less active contributors tended to use more collective team processes, communicated the most about leisure and expressed higher amounts of cognitive processes. In particular, those that were less active used more insightful processes; these members expressed significantly more tentativeness and they also communicated with more certainty. Collective team processes are an indicator of team synergy (Tuckman, 1965). Denning (2012) noted that collective language use is a sign of team focus, and so may be good for collaborative teams. These findings reveal that the less active IBM Rational Jazz practitioners operated cohesively in the norming and performing phases of group work (Tuckman, 1965). Generally, teams tend to engage more collectively after overcoming initial differences and conflicts, and elevated levels of collective behaviours are an indicator of more shared and established team norms (Tuckman, 1965). This reasoning is applicable to the IBM Rational Jazz teams studied here, especially given that the core developers were found to use lower levels of individualistic processes and those that were less active used higher level of collective language. Results for leisure also showed that IBM Rational Jazz's less active practitioners communicated significantly more of this form of language when compared to the core developers (refer to Section 4.2.1). Those less active also communicated with higher levels of cognitive processes (refer to Section 4.2.1); cognitive qualities have been linked previously to higher software task performance (Andre et al., 2011).

This evidence endorses the previous assessment of the way core developers maintained task focus; perhaps core developers' attentiveness on work and achievement was in part driven by their desire to keep their teams focused when the less active members became highly social and engaged excessively about leisure. At the same time, those that were

less active also contributed meaningfully with ideas, and expressed certainty during their contributions (refer to Section 4.2.1). These findings increment those that were revealed in Section 5.1.3 about the importance of the less active members; and particularly, the observation that these members were highly interconnected.

Overall, these behavioural processes may have a balancing effect (Licorish & MacDonell, 2012), especially in terms of IBM Rational Jazz teams' self-organisation. Benne & Sheats (1948) have shown that various roles (both social- and task-based) are acted out by contributors and are necessary to maintain team balance during successful team work. Social roles contribute towards positive group climate, promoting harmonizing and compromising traits, while task roles are concerned with task success, contributing and initiating ideas and knowledge towards task completion. Evidence in this work suggests that, as a group, IBM Rational Jazz teams indeed cultivated both social and task oriented team norms (Chang et al., 2013). This finding would seem to be generally beneficial for globally distributed agile software developments, and likely impacted positively on IBM Rational Jazz teams' trust, and overall performance (Dullemond et al., 2009; Lee & Yong, 2010).

In keeping with the above discussion, Section 4.2.1 shows that, overall, IBM Rational Jazz teams did not use high amounts of negative language. Excess negative emotion may lead to disharmony and hostility among teams (De Dreu & Weingart, 2003). Those that demonstrate negative team traits may be annoyed or irritated (Denning, 2012), and evidence of this trait is generally understandable during periods of schedule slippage, as a result of the discovery of defects in code that had undergone seemingly rigorous testing, or for other external organisational effects (e.g., inter-departmental disagreements or due to changing client requirements). However, in extreme cases such negative traits may also be deep-seated in anger and resentment, which is likely to result in an individual's desire to undermine their team's vision due to their own personal dissatisfaction (Solomon, 2007). Thus, negative moods are – unsurprisingly – generally bad for teamwork (Denning, 2012; Goguen, 1993; Goldberg, 1981).

Research has shown that team moods, and particularly those related to collective, social and encouraging processes that support teamwork and optimism, generally promote team satisfaction and cooperation, and these behavioural processes have a positive influence on team morale and task outcomes (Denning, 2012). A social team ambience has also been shown to encourage team members' contributions and rapport (Chang et

al., 2013). Such positive moods are thought to be cultivated by skilled team leaders (Denning, 2012).

In summary, this section represents a first attempt to analyse and discuss core developers' attitudes and to compare the traits that these members exhibited to those of their less active counterparts. Apart from triangulating the findings in Section 5.1 above, discussions in this section were also aimed at extending previous works that had discovered many software teams to operate in centralised communication and task performance structures. It was observed that core developers exhibited less individualistic attitudes, these members delegated more, and were largely responsible for maintaining task focus. These traits were assessed as being critical for maintaining desirable team norms, and particularly, given core members' central position in their teams, it was noted that these traits are likely to have a positive impact on the attitudes that are cultivated by the wider IBM Rational Jazz teams. These findings were somewhat in contradiction to those discussed in Section 5.1.3 which did not find core developers to be significantly more important than their less active counterparts, an issue that reflected the limitation of the frequency-based technique (the SNA closeness measure) that produced this evidence. It was observed that the less active developers complemented the core developers, and also helped to maintain team balance. Given that most of the core developers were informal leaders, results in this work also suggest that a flexible organisational climate and less rigid project management are likely to be advantageous for globally distributed agile software team management. Additionally, findings discussed in this section imply that core developers were intrinsically driven. These discussions are extended in the following section (Section 5.2.2) where the roles core developers enacted are examined through more contextual lenses.

5.2.2 Enacted roles (RQ7)

In an effort to study IBM Rational Jazz practitioners' behavioural processes and to compare the behaviours of core developers against those of their less active counterparts so as to understand the true role of core developers, beyond the communication patterns noted previously (Crowston et al., 2006; Mockus et al., 2002; Shihab et al., 2009), practitioners' messages were analysed using linguistic analysis in Section 4.2.1. These results were discussed in the preceding section (Section 5.2.1), which shows that core developers communicated significantly different attitudes to the other members of their project. Such attitudes are largely responsible for maintaining a team perspective and

for promoting task and achievement focus in the team. Less active Jazz practitioners used high levels of collective and social processes, and these members were also more insightful. It was contended that these findings are fitting for globally distributed software developments given core developers' central position in their teams' communication networks, and the highly connected nature of IBM Rational Jazz project's networks in general. These discussions are incremented here. This section discusses the contextual directed content analysis results presented in Section 4.2.2 to evaluate the core developers' enacted roles in their teams. These discussions follow a similar outline to those in Section 5.2.1, where core developers' interactions and their enacted roles are compared to those of their less active counterparts. This approach is taken to understand further how core developers (and others) contribute to their team dynamics.

Given the high volume of messages conveyed by core developers as well as their intensive involvement in task changes (refer to Section 4.1.3 and Section 5.1.4), it was anticipated that these individuals would dominate knowledge sharing in their teams, and the results in Section 4.2.2 support this position. Here it is shown that IBM Rational Jazz core developers contributed 42% of their teams' actual utterances (note that these members also communicated 46% of their teams' messages – refer to Section 4.2.1). These results show that core developers' communications were aimed at much more than task coordination alone (Cataldo & Herbsleb, 2008). This finding is revealing when considering that the core developers in the three project areas that were selected for the directed content analysis procedure (P1, P7 and P8 – refer to Table 2) comprised only 6 of their teams' 107 members (being two programmers, three team leads and one project manager). In particular, as noted in Chapter 3, a person occupying the formal 'Programmer' (contributor) role in IBM Rational Jazz is defined as a contributor to the architecture and code of a component, the 'Team Leader' (component lead) is responsible for planning and architectural integrity of the component and the 'Project Manager' (PMC) is a member of the project management committee overseeing the Jazz project. Additionally, each IBM Rational Jazz team is led by a formally appointed project manager.

Furthermore, this finding regarding the very high level of core developers' utterances represents a substantial difference in involvement – it is 45 times the average knowledge contribution of a typical Jazz practitioner (refer to Section 4.2.2). This finding supports those discussed in Section 5.1.2 above, and confirm that developers in

distributed global software developments are not required to contribute equitably in order for the team to succeed (Crowston et al., 2006; Mockus et al., 2002; Shihab et al., 2009). Additionally, the directed content analysis results (as well as the linguistic analysis results discussed in the preceding section – Section 5.2.1) also refute the assessment that core developers are not significantly more important than the less active members, and in doing so support the need for triangulating quantitative analysis techniques with deeper approaches when studying team behaviours. These results also endorse those that were presented in Section 4.1.3 that core developers' active involvement in communication is perhaps driven by their task performance (Bird et al., 2006a; Cataldo & Herbsleb, 2008; Shihab et al., 2010).

In examining the actual details of core developers' utterances (refer to Section 4.2.2), it is shown that these members indeed operated across multiple informal roles. The directed CA results confirm that core developers were integrally involved with team organization and task assignment (e.g., see measures for Answers and Instruction in Figure 25 and Figure 26). It had been previously established that individuals involved in such forms of (vertical) communication are seen as capable, and such individuals are often perceived by their peers as knowledge hubs, and pillars of the knowledge construction process (Henri & Kaye, 1992; Zhu, 1996). Discourses of an assertive nature (e.g., Type II Questions and Instructions) are also communicated due to a perception that little authoritative feedback is forthcoming (Zhu, 1996), and may generally be linked to those in power. In fact, such responsibilities and behaviours are often associated with formal software project leadership or individuals occupying more coordination and planning related roles (Andre et al., 2011). These findings converge with those that were discovered during the linguistic analysis regarding the higher level of delegation language that was communicated by core developers when compared to their less active counterparts (refer to Section 4.2.1). Additionally, evidence revealed in this work suggests that these globally distributed agile software teams promoted organic and informal work structures in order to self-organise (Hoda et al., 2010b). Thus, formal role assignment may not be an adequate indicator of the need for communication and coordination during globally distributed software projects (Acuna et al., 2006; Andre et al., 2011).

Results in Section 4.2.2 revealed that core developers provided context awareness for the other team members and acted as their teams' main information resource (e.g., as evident in the measures for Information sharing, Discussion and Scaffolding in Figure

25 and Figure 26). Such competencies are typically associated with highly skilled roles; or with those individuals that are extremely creative, imaginative and insightful (Belbin, 2002). Those that communicate more are also generally more aware (Kanawattanachai & Yoo, 2007; Palazzolo, Serb, She, Su, & Contractor, 2006), as demonstrated with the level of Information that was shared by core developers. Such vigilant and aware individuals are necessary for globally distributed agile developments (Al-Ani et al., 2011; Young & Terashima, 2008). These findings coincide with those for task changes (refer to Section 4.1.3), denoting that core developers were indeed their teams' main implementers. Importantly, while core developers were actively involved with their teams' task portfolio, these members also exhibited intrapersonal and interpersonal skills (e.g., see measures for Apology, Off task and Gratitude/Praise in Figure 25 and Figure 26) (Downey, 2009). These observations triangulate the low levels of individualistic language communicated by core developers and confirm that Jazz core developers' active involvement in their teams' communications was fitting for promoting shared team norms (Chang et al., 2013).

Core developers also communicated with less desired judgemental language (e.g., see measures for Comment in Figure 25 and Figure 26) on occasions. Lower incidence of judgmental discourse is often required for maintaining team spirit and overcoming tension, which is critical to a positive team atmosphere (Belbin, 2002; Benne & Sheats, 1948). However, while excessive debate and conflict behaviours are posited to be harmful, some level of task-related conflict is also said to be good for enhancing innovativeness and critical evaluation among group members (Tjosvold, 2008). Perhaps in their drive for maintaining strong task performance, core developers became evaluative and judgemental at times. While the evidence obtained from this static project analysis does not reveal specifically when these judgemental behaviours are most prevalent for core developers or why, such understandings (as provided in Section 5.3.2) would be useful for implementing strategies aimed at discouraging excessive expression of such attitudes, which may derail team cohesiveness (Benne & Sheats, 1948; Denning, 2012).

In terms of the less active members, apart from their minimal involvement in providing Instructions, these members also supported core developers and contributed to all the interaction categories, albeit in much smaller amounts (refer to Section 4.2.2). Overall, the evidence presented in Section 4.2.2 suggests that the formal project managers in these IBM Rational Jazz teams acted as facilitators, and were happy to let their teams

self-organize, an approach often deemed necessary for agile teams to succeed (Abrahamsson et al., 2003). Such a hands-off approach to project governance may only be feasible (and useful for globally distributed software teams) if team members are achievement motivated and informal leaders are present – the core developers.

As noted above (refer to Section 5.1.5) the behaviours demonstrated by IBM Rational Jazz core developers may not be default behaviours, however. Such high performing members often need to possess intrinsic motivation and keen willingness to self-organize (Kline & Peters, 1991; Moe, Dingsoyr, & Dyba, 2008). A facilitating organization and work structure may also encourage high performers to work across roles as the need arises. On the basis of the result discovered in Section 4.2.2 it is further contended that IBM Rational Jazz is one such organization that encourages team members' performance based on their natural abilities. Such institutional configurations enable team members to adapt and execute their tasks based on work demands (Powell, 1990), as was evident among Jazz practitioners. Such arrangements should be encouraged in order to facilitate globally distributed software team success.

As a group, IBM Rational Jazz developers communicated extensively about actual software development, and these practitioners were highly task focussed. Over 95% of Jazz practitioners' exchanges related directly to solving software tasks (refer to Section 4.2.2). These findings suggest that core developers indeed drove their teams' task focus, and all Jazz developers were happy to contribute towards their teams' performance. These outcomes may have implications for tool design, and particularly, for tools aimed at prioritising team communications. Overall, while IBM Rational Jazz developers were highly task oriented, Jazz practitioners also expressed some amounts of gratitude and communicated modestly about personal issues. Such an interpersonal outlook confirms the previous linguistic results for IBM Rational Jazz teams' high level of social and positive processes (refer to Section 5.2.1). In fact, although IBM Rational Jazz practitioners used social and positive language processes, the contextual directed CA results (refer to Section 4.2.2) revealed that these individuals maintained most task focus. These findings suggest that while a positive and social group atmosphere may be ideal for maintaining a pleasant team ambience (Chang et al., 2013; Denning, 2012), a task-driven team focus (and roles) is important for globally distributed agile software developments. Previous literature has endorsed this viewpoint for collocated settings (Andre et al., 2011), suggesting that a task-driven team outlook is appropriate for software development (and team performance) as a whole.

5.2.3 Summary

In an effort to understand the true role of core developers, so as to provide details around *the reasons for* these practitioners' distinct presence in communication and task performance and their contribution to team dynamics, Section 4.2 diverges from the more commonly used quantitative and frequency-based analysis approaches and has instead used deeper psycholinguistic and directed CA techniques in the study of IBM Rational Jazz teams' messages. These findings were discussed in this section.

In summary, the linguistic analysis results show that, when compared to the less active software practitioners, core developers expressed little self-focused attitudes, they delegated more, and they were highly task-driven (refer to Section 5.2.1). Evidence of the low amount of self-focused attitudes and high levels of task-driven behaviour for core developers is ideal for globally distributed software teams, where the need for collective team vision and an outcome-oriented viewpoint are critical to positive team performance. In particular, given core developers' influential position in their teams, these behaviours are no doubt useful for encouraging desirable team norms. The high level of delegation language also endorses the view that core developers operated as their teams' leaders (these results extend those in Figure 29, and are depicted in the orange segments of Figure 30). Given that some core members did not occupy formal leadership roles, this evidence also confirms that IBM Rational Jazz, as an organisation, promoted organic work structures and encouraged participation based on practitioners' competencies. Such an approach to software project governance is likely to be critical in order for globally distributed software development teams to succeed, and particularly for supporting highly motivated and keen performers.

Jazz's less active members contributed the most collective and social processes and were also heavily involved in their teams' cognitive processes (refer to Section 5.2.1). These attitudes complimented the task-driven traits expressed by the core developers, and suggest that, overall, IBM Rational Jazz practitioners operated in a cohesive and well-balanced team environment. Such an environment (possessing both social- and task-based individuals) is posited to be necessary for high performing teams, and particularly for globally distributed software developments.

The contextual directed content analysis results reveal that core developers dominated their teams' knowledge processes, denoting that globally distributed software team members are not required to perform equitably to be successful (refer to Section 5.2.2).

Additionally, findings for the significant level of knowledge contributions by core developers support earlier propositions that these members were very important to their teams, and their involvement in communication may be driven by their task portfolio. Results from the directed content analysis also show that core developers indeed operated across informal roles (refer to Figure 30 for illustration – note that the orange aspects in this figure extend the work's outcomes in Figure 29). For instance, core developers were integrally involved with team organization, task assignment and project instructions. Such responsibilities are typically assigned to the formal project manager; however, only one of the core developers belonged to this role. These findings converge with those for the high level of delegation language expressed by core developers, as noted in the linguistic analysis results.

Contextual analysis also shows that core developers were their teams' main information resource, and these practitioners were particularly insightful. Furthermore, it was confirmed that core developers were their teams' main solution providers. That said, core developers also expressed intrapersonal and interpersonal traits. These findings triangulate those for core developers' task-focused attitudes and their low level of individualistic behaviours. Overall, directed CA evidence confirms that core developers' informal responsibilities exceeded what would be typical of their formally assigned roles. Support for the way core developers operated across roles endorses the viewpoint that the IBM Rational Jazz organisation facilitated teams' self-organisation, and indicates that such an approach is relevant to successful globally distributed software project governance.

In terms of the less active developers, directed CA findings show that although these members contributed modestly to their teams' communication networks and task performance, these practitioners were also important to their teams' output. These individuals contributed insights, and were key to their teams' interpersonal climate. As with the linguistic analysis results, directed CA findings confirmed that IBM Rational Jazz teams operated in a balanced team environment. Additionally, although both social- and task- driven behaviours were prevalent among IBM Rational Jazz teams, these practitioners maintained most task-focus. The need for task focus was previously discovered in co-located software development settings, suggesting that such behaviours are necessary for software development (and team performance) as a whole.

The psycholinguistics and directed content analysis evidence exposed the limitations inherent in frequency-based SNA techniques in terms of studying the internal details of software practitioners' behaviours and team dynamics. Evidence revealed from these techniques (linguistics and directed CA) extended previous work and provided further explanations for the reason for the centralised patterns previously noted for software teams' communication networks, and details around core developers' true roles.

Of final note here is the evidence that core developers expressed a degree of less desirable judgemental behaviour. While a modest level of this form of attitude is helpful for promoting innovativeness and critical evaluation among group members, large amounts of such behaviours are negative and create team tension. Given core developers' influence in their team communication, it is pertinent to understand the circumstances under which these and similar behaviours are revealed. Various events may impact the attitudes and knowledge behaviours expressed by core developers which may in turn impact their overall teams' behavioural climate, and team outcomes. Thus, an understanding of core developers' attitudes and the knowledge sharing behaviours that they contribute through the course of their project would be useful for informing strategies aimed at maintaining an optimistic and positive team climate, and ultimately, positive team performance. More generally, such knowledge would reveal details around *when* core developers are more or less likely to perform as desired, during which project phases these practitioners are most influential, and when their teams are most likely to benefit from their knowledge and experiences. Additionally, these insights would be useful for understanding the specific traits of less active team members that are likely to complement these individuals. Furthermore, such understandings would inform specific project arrangements that are likely to enhance the satisfaction of core developers and inform project governance, and team strategies aimed at composing software teams in readiness for core developers' less productive periods. These understandings are provided in the following section (Section 5.3), where the ways in which core developers' attitudes, knowledge sharing behaviours and task performance change over the course of their project are discussed.

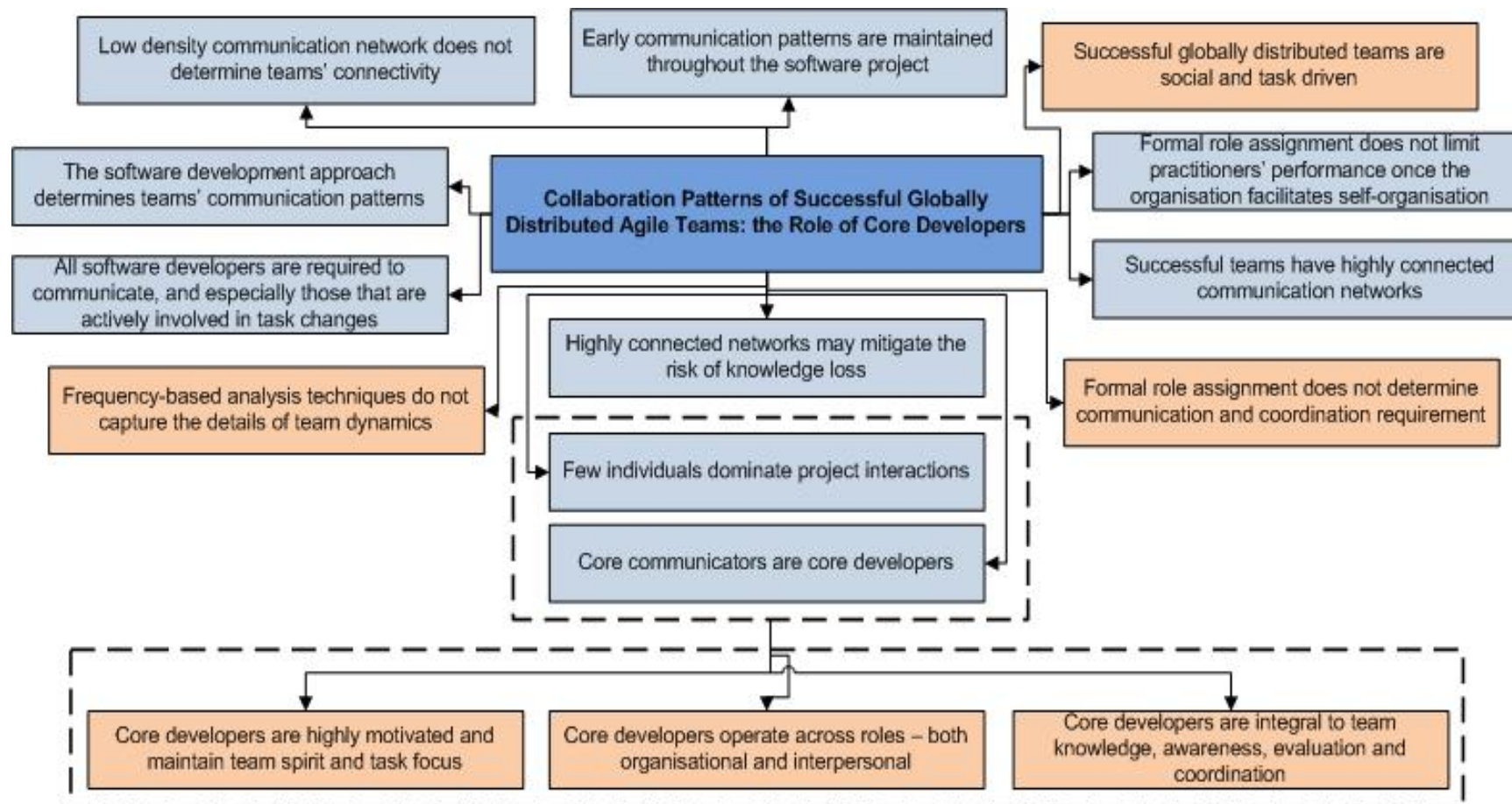


Figure 30. Collaboration patterns of successful globally distributed agile software teams and the true role of core developers

5.3 Changes in core developers' attitudes, knowledge sharing and task performance (Phase 3)

In line with the previous findings of Bird et al. (2006a), discussions above confirm that core developers communicated the most in the Jazz project, and those that had frequent discourses were also integral to their teams' actual software development portfolios (Cataldo & Herbsleb, 2008) (refer to Section 5.1). Overall, it was observed that Jazz core developers demonstrated all specialties, including high levels of interpersonal, organizational and social skills (refer to Section 5.2). Results in Section 4.3 suggest, however, that core developers' specialties were enacted in different ways during different project phases, tending to align with the teams' needs and their involvement in task performance. This section discusses these results (those in Section 4.3), and examines the changes in core developers' efforts over the duration of their project by considering RQ8 – RQ13 in turn.

First, Section 5.3.1 discusses the way core developers' attitudes change over their project and answers RQ8 (Do core developers' attitudes change as their project progresses?). Section 5.3.2 then provides a discussion around the way core developers share knowledge over their project, in order to address RQ9 (How do core developers share knowledge over the course of their project?). RQ10 (What initial team arrangements lead to developers becoming hubs in their teams?) is discussed in the third section (Section 5.3.3), and the fourth section (Section 5.3.4) discusses RQ11 (How do core developers contribute to task performance over their project?). The fifth section (Section 5.3.5) presents a discussion of the way core developers attitudes are linked to their task performance and so answers RQ12 (Are core developers' contributions to task performance linked to their attitudes?). Section 5.3.6 then considers the discussion for the final question, RQ13 (Are core developers' contributions to task performance linked to their contribution of knowledge?). Finally, a synopsis of the discussions that are presented throughout this section is provided in Section 5.3.7.

5.3.1 Changes in attitudes (RQ8)

For the most part IBM Rational Jazz core developers communicated consistent attitudes over the duration of their project, but they were most cognitive at project initiation, and exhibited the most individualistic and negative attitudes when they were least involved in task (WI) changes. Overall, certain attitudes were more pronounced in specific project phases. For instance, Jazz core developers became more self-focused as their

project progressed, but these individuals also exhibited high levels of collective attitudes over their project. As noted in Section 5.2.1, those exhibiting individualistic behaviours are said to affect team spirit and these traits have a negative effect on team cohesion (Benne & Sheats, 1948), while the opposite is shown for those that are collective (Pennebaker & Lay, 2002). It is contended that these practitioners' heavy development portfolios in the middle project phases and the challenges of release pressures towards project completion were responsible for the higher levels of self-focused behaviour at these times. The rigours associated with release pressures have been shown previously to affect team optimism (Rigby & Hassan, 2007). In the Jazz context, however, core developers communicated the least at project completion, and so the higher levels of self-focus may have minimal negative impact on their teams at this time. This form of attitude may not be desirable during the early-mid and late-mid phases, when core developers are most dominant, and thus, are highly likely to affect overall team climate and trust (Dullemond et al., 2009; Lee & Yong, 2010). A project manager observing these trends should be particularly vigilant and encourage active contributions from the wider team during these times, as core developers generally occupy the centre of coordination action and their teams' knowledge processes (Leavitt, 1951) (refer to Section 5.1 and Section 5.2). Otherwise, core developers' self-focused attitudes could potentially affect overall team morale in a negative way. Given the reduced opportunities for informal (and face-to-face) communication in AGSD, such attitudes may also strain team trust.

Core developers expressed the most cognitive attitudes at the start of their project, at a time when project features were being initiated. These findings are positive, given the need for project leaders to be perceptive and insightful during early project scoping activities. Such knowledge may become critical to team awareness, and later efforts related to project oversight and monitoring (Damian & Zowghi, 2003; Prikladnicki et al., 2003; Rudzki et al., 2010), which are often challenged by distance. The wider Jazz team is also likely to benefit from these practitioners' higher levels of insight at this time and so it would be prudent for project managers to implement strategies aimed at encouraging the simultaneous engagement of less active developers at project inception. Results in Section 4.3.1 show that while Jazz core developers were highly task-focused (Benne & Sheats, 1948) overall, these attitudes were most evident in the latter phases of their project. These findings are particularly informative for the end phase given core developers' lower levels of communication and task performance at this time. As noted

previously, while project release pressure is likely to promote such urgency among the core developers, this finding supports the view that these practitioners were also motivated to see their project through even when they were not entirely in control. As noted in Section 5.2.1 such strong team commitment is linked to a sentimental attachment to team goals (Allen & Meyer, 1990; Morgan & Hunt, 1994), and is particularly fitting for overall team performance.

Relatively speaking, core developers were also most negative and cynical towards project closure. These findings coincide with the results for the expression of individualistic and self-focused attitudes in Section 4.3.1. Like individualistic attitudes (Pennebaker & Lay, 2002), negative behaviour is counterproductive for team work (Goldberg, 1981) (noted in Section 5.2.1). However, positive and social language use is an indicator of team friendliness (Benne & Sheats, 1948), and core developers also used significant amounts of these forms of language throughout their project (although these processes were less evident in the last project phase). Individuals that are motivated and operate in environments with positive social ambience are happy to contribute (Chang et al., 2013). These forms of language were especially pronounced in the early phases for core developers, a time when interpersonal skills are critical to team formation and establishing team dynamics (Downey, 2009). Given the likely establishment of positive team norms between core developers and the rest of their teams in the early project phases, the wider team may be more tolerant of core developers' frustrations at project completion. These higher levels of social and positive processes may also offset the more cynical attitudes. However, as noted above, keen participation (including clear communication) and availability of the wider team may go some way to reducing tension and coordination breakdowns (Herbsleb & Roberts, 2006) and enhance team spirit (and trust) around project completion for core developers. These discussions are extended in the following section (Section 5.3.2), where core developers' changes in knowledge sharing behaviours over the course of their project are examined from a directed CA perspective.

5.3.2 Changes in knowledge sharing (RQ9)

While core developers exhibited higher levels of some specific knowledge sharing behaviours at project start and completion, overall, these members shared more knowledge in the middle phases of their project. Additionally, the more software development IBM Rational Jazz core developers performed, the more they were

required to provide knowledge to their teams. These findings support this proposition, as put forward in Section 5.1.4 and Section 5.2.2. The contextual results in Section 4.3.2 show that core developers were integrally involved in their project (Licorish & MacDonell, 2013c), and that Jazz core developers were most dominant during the middle phases of their project. Additionally, these practitioners asked more questions at project start and project completion. The high level of questioning at project initiation is understandable given the need to clarify and delineate system requirements and team understandings at this time. The elevated level of questioning at project completion coincides with core developers' reduced levels of project communication and task participation. The questions at this time may be aimed at the need for context awareness, given this reduced overall involvement. This finding supports the view that core developers were anxious for their project to succeed even when they were not so heavily engaged (also posited in Section 5.3.1). Highly curious individuals are not able to manage their desire for discovering new knowledge (Denning, 2012), and these individuals generally tend to make many inquiries.

Findings for Answers in Section 4.3.2 were the reverse; it was observed that core developers provided most answers to their teams during the middle phases, when they were most active. These results demonstrate that although Jazz core developers were focused on their individual tasks they also kept the teams' agendas in mind (Belbin, 2002; Pennebaker et al., 2003; Tuckman, 1965). This assessment is supported by the other knowledge dimension analyses (refer to Section 4.3.2) – this pattern of involvement is maintained for core developers' contributions of Information, ideas (Discussion) and suggestions (Scaffolding) to their teams, which were most frequently provided during the middle phases of the project.

In fact, results for all of the other knowledge measures (except Reflection which did not follow a consistent pattern) demonstrate that core developers were indeed most active during the middle phases, when they were also highly involved in task changes. In particular, core developers expressed most judgemental behaviours (Comments) during the middle project phases. Although excessive debate and judgmental behaviours are posited to be harmful, as noted earlier in Section 5.2.2, some level of task-related conflict is also said to be good for enhancing innovativeness and critical evaluation among group members (Tjosvold, 2008). Additionally, given that core developers were most judgemental when they were most active in their teams, these behaviours may not necessarily present a threat to overall team performance. In fact, it was generally

expected that core developers would express such attitudes when they were not involved in task (WI) changes due to frustration (as was noted for negativity in Section 5.3.1). However, the evidence revealed in Section 4.3.2 does not support this assessment. These findings indeed suggest that critical evaluations may aid task performance.

Support for the view that the higher level of debates that were contributed by core developers during the middle phases of their project may be more evaluative than cynical is seen in the findings for the expression of Gratitude and Off task utterances. Although core developers did not communicate heavily about personal issues, Section 4.3.2 confirms that core developers communicated socially (Off task) and expressed more gratitude when they were most actively involved in solving software tasks.

Overall, these findings show that technical needs drove IBM Rational Jazz teams' knowledge sharing (Cataldo & Herbsleb, 2008), such that task involvement appears to have a general influence on core developers' need to provide knowledge and information to their teammates. Such an understanding was established by an early study considering human factors during software development (Curtis, 1981). Thus, communication hubs may not necessarily be formally denoted as communication and coordination specialists (Cataldo & Herbsleb, 2008). Rather, the findings in this work indicate that these individuals communicate because of their actual development portfolio. While the presence of such individuals in teams is no doubt beneficial, project managers should also be vigilant about the team's possible over-reliance on these members, which may negatively affect the quality of the knowledge core developers are able to provide. Poor quality information may lead to irreparable damage in distributed development settings, particularly due to reduced opportunities available for (re)orientation to maintain the shared understanding (Prikladnicki et al., 2003; Tiwana, 2004).

This evidence of the way core developers provide knowledge to their teams and their general interest in team performance may be linked to their teams' or tasks' demands (Hackman, 1986). However, previous work has also found that such behaviours may be intrinsic (Belbin, 2002; Chang et al., 2013; W. Oh et al., 2006). This issue is examined further in the following section (Section 5.3.3).

5.3.3 Becoming team hubs (RQ10)

IBM Rational Jazz core developers exhibited cognitive competencies at project initiation; however, the position these practitioners occupy in their teams is linked to their active involvement in task changes. This involvement drives the need for their distinct presence in their teams' knowledge sharing networks. It was revealed in Section 4.3.2 that core developers asked more questions when they were least involved in task changes, and these practitioners were required to provide more help when they made more task changes. During periods of core developers' reduced levels of involvement their less active counterparts made substantial contributions. As a result, there was a need for 're-orientation' of core developers regarding these activities as evidenced by the lower levels of task involvement and higher proportion of questioning towards project completion (refer to Section 4.3.2 and Section 4.3.3). The need to await feedback is likely to promote uneasiness and frustrate core developers. This issue may not be easily avoided in globally distributed agile software development settings, however. Thus, access to multiple communication channels is important in reducing such dissatisfaction (Damian & Zowghi, 2003). The availability of a variety of communication channels would also be beneficial to support reflections conducted at project completion (refer to Section 4.3.3), comprising information that will likely be relevant to future developments.

Results in this work show that, regardless of their role title, Jazz core developers continuously maintained awareness of and were anxious regarding their teams' overall progress (refer to Section 5.3.2 for further details). Additionally, while core developers delegated, they were also happy to provide answers and help to their peers. This level of project ownership is especially surprising given that more than 50% of the core developers in this study occupied the nominal 'programmer' role (refer to Section 4.1.3). Cataldo and Herbsleb (2008) also found non-lead team members at the centre of project communication and task changes. These findings indicate that IBM Rational Jazz core developers were highly skilled and motivated in their teams (Sach et al., 2011). Findings show that core developers provided more knowledge to their teams when they made higher levels of task changes. Thus, the high levels of team reliance may in part be responsible for core developers' distinct presence in their project's communications. However, these individuals also expressed high levels of cognitive traits (Feldt et al., 2010) at project initiation suggesting that they possessed some unique insightful characteristics, and these were intrinsic.

Core developers' contributions to task changes have been noted thus far in Section 5.3.1, Section 5.3.2 and in this section (Section 5.3.3). However, references to this subject were generally made to support the assessment of other issues (e.g., the way core developers share knowledge over the course of their project). This issue is now properly examined in the following section (Section 5.3.4).

5.3.4 Changes in task performance (RQ11)

IBM Rational Jazz core developers' involvement in task performance increased steadily over the first three project phases, and decreased towards project completion. In Section 4.1.3 it is shown that, overall, the core developers in this study initiated more than 41% of their teams' software tasks, they made more than 69% of the changes to these tasks, and resolved nearly 75% of all software tasks undertaken by their teams (Licorish & MacDonell, 2013c). These findings are revealing when considering that core developers comprised just over 10% of their teams' 146 practitioners (refer to Section 4.1.3). In Section 4.3.3 results for core developers' involvement in task performance over the course of their project substantiate the project snapshot results. As noted above, IBM Rational Jazz core developers were most active in the middle phases of their project. In particular, these practitioners made the highest number of task (WI) changes in the late-mid stage of their project, with task changes tending to be stable, and much lower, during their project's start and end phases. Overall, core developers contributed the fewest number of task changes towards project completion. While this evidence was previously linked to the increasing level of task difficulty encountered as software projects progress (Cataldo & Herbsleb, 2008), the trend of the results over the first three project phases in this work does not support this position. It may instead be contended, given the reduction in communication (refer to Table 17), that core developers were involved in other work outside of the repository towards project completion. Future research employing complementary interview-related techniques would help to provide additional insights into the reasons for such a pattern.

The patterns noted for core developers' involvement in task changes may be linked to their attitudes. In particular, when these members express specific moods they may be more or less inclined to perform. Various events may modify team atmosphere which may result in both negative and positive team outcomes (Rigby & Hassan, 2007). Such knowledge would be useful for informing software project governance. This issue is considered next in Section 5.3.5.

5.3.5 Attitudes and task performance (RQ12)

Core developers were most actively involved in task changes when they were work focussed. However, overall, results in Section 4.3.4 do not provide a conclusive link between core developers' attitudes and their contribution to software tasks. Apart from work related focus, there were some linkages between task (WI) changes and the expression of some other specific behaviour.

For instance, although a small negative correlation between core developers' use of collective language and the number of changes they made was discovered in Section 4.3.4, this result was not significant. While it would be rational for core developers to express more collective attitudes (Inkpen & Tsang, 2005; Levin & Cross, 2004; Pennebaker & Lay, 2002) when relying on others for help (and this was evident to an extent in the results), this evidence was not strong. Opposite patterns of results were observed for the cognitive attitudes. When core developers communicated higher amounts of insightful language, they made more task changes, and when these individuals communicated with higher levels of discrepancy language they made fewer task changes (De Vries et al., 2006). Insightful language comprises words such as "think", "consider", "determine", and "idea" (Pennebaker & King, 1999). It could be expected that there would be elevated use of such words when assistance is being provided (e.g., "you should consider using session variables instead of cookies to maintain state across the web pages" or "I think the bug you are now noticing was observed after last night's build"), as was seen for core developers in the early-mid and late-mid phases of their project. Thus, this evidence provides confirmation of the results that core developers were involved in higher levels of knowledge sharing when they made more task changes (De Vries et al., 2006).

Discrepancy language comprises words such as "should", "prefer", "needed", and "regardless" (Pennebaker & King, 1999). Such words are likely to be used to offer suggestions (e.g., "the patch I created for bug B should also work for bug C") or to show preference for a specific option (e.g., "I prefer option E over option F"). Thus, the finding that when core developers used higher amounts of discrepancy words they made fewer task (WI) changes can be explained. However, overall, results in Section 4.3.2 were not definitive; this question would benefit from additional research, comprising a much larger sample of artefacts.

That said, given the correlation results for work-related language use and task performance in Section 4.3.4, profound use of this form of language by core developers may be a signal for the less active practitioners to reduce their level of reliance on these members. Additionally, evidence of this form of language process may also result in the provision of support for core developers, should they need such assistance. Furthermore, excessive use of work-related utterances may also stimulate project managers' interest, and serve as a warning sign that project leaders should be vigilant about the quality of the feedback that core developers are able to provide the general team at this time (noted in Section 5.3.2). These results are triangulated with those that are discussed in the following section (Section 5.3.6).

5.3.6 Knowledge sharing and task performance (RQ13)

IBM Rational Jazz core developers' performance in task changes is related to their contribution of knowledge (Geen, 1991; Inkpen & Tsang, 2005; Levin & Cross, 2004). It was observed that when there were higher levels of questioning from core developers these practitioners were less active in task changes, tending to rely more on their wider teams. Thus, strategies aimed at surrounding core developers with competent communicators would help these practitioners to quickly become familiar with task knowledge. This would in turn promote overall shared team understanding (Boh, Slaughter, & Espinosa, 2007), and reduced incidence of coordination breakdowns (Herbsleb & Roberts, 2006). This may be particularly useful for team synergies in globally distributed developments, as while it is not entirely clear from the results that were revealed in this work why core developers demonstrated reduced presence at specific times of their project (refer to Section 4.3.3), it was evident that these individuals expressed more unhappiness when they communicated many questions. This finding suggests that they were indeed dissatisfied with the feedback they received at times (Goldberg, 1981). In contrast, when core developers were actively involved in task changes they provided more answers to their wider teams (De Vries et al., 2006). A similar pattern of results was revealed for Information sharing, Discussion, Scaffolding, Instruction and Comments. Use of these forms of communication was related to core developers' active involvement in task changes (Bock & Kim, 2002).

Overall, the results presented in Section 4.3.5 confirm previous psycholinguistic theories (Mairesse et al., 2007; Pennebaker & King, 1999) and support the contention that there is evidence of individuals' behaviours and attitudes in their communications

(Pennebaker et al., 2003), and that word use has a deeper link to individuals' behaviour (Bales, 1950b; Geen, 1991; Quigley et al., 2007). For example, it was established that when IBM Rational Jazz practitioners used words such as "think", "consider", "determine", and "idea", they shared more ideas, offered more suggestions and were involved in more critical evaluations – and did more actual software development. These findings have implications for software engineering research and practice, considered in the next chapter (Chapter 6).

5.3.7 Summary

In this section core developers' attitudes, knowledge sharing behaviours and task performance were considered over the course of their project. First, Section 5.3.1 discussed the changes in core developers' attitudes as their project progresses. It was noted that core developers' attitudes did not vary extensively over their project. However, these practitioners were most cognitive at project initiation, and exhibited the most individualistic and negative attitudes when they were least involved in task (WI) changes towards project completion. On the other hand core developers exhibited social and positive attitudes throughout their project. The reduced level of project involvement for core developers at project completion may result in increased likelihood of disharmony among their teams, given the higher incidence of less desired attitudes at this time. This is counterproductive for teamwork, and particularly for global teams given their reduced opportunities to build and maintain team trust. However, it was noted that active participation of the less active members towards project completion would likely mitigate the possible negative effect of core developers' unhappiness. Additionally, the consistent expression of social and positive attitudes may also offset the negative attitudes that were expressed by core developers. Core developers' higher level of cognitive processes at project initiation may also benefit the less active team members. Furthermore, core developers were highly task focused even when they were not heavily involved in task changes, an indicator that these individuals were indeed intrinsically driven.

Section 5.3.2 next discussed the way core developers share knowledge over the course of their project. Core developers exhibited higher levels of some specific knowledge sharing behaviours at project start and completion; however, overall, these members shared more knowledge in the middle phases of their project. These practitioners asked more questions when they were least involved with task changes and provided more

answers and overall knowledge to their teams when they were most active. These contributions also comprised judgemental utterances, which, given the corresponding evidence of off task communication and gratitude contributed by core developers, may generally be useful for critical evaluation of others' suggestions. Given the core developers' extensive involvement in their teams' knowledge processes throughout their project, project managers are encouraged to be vigilant about the actual quality of the knowledge these members are able to provide.

Section 5.3.3 considered the initial team arrangements that lead to core developers becoming hubs in their teams. It was noted that core developers' actual position in their team was driven by the number of task changes they made, which in turn influenced their teams' dependence for contextual awareness. However, core developers also exhibited insightful competencies from very early in their project. These discussions were supplemented by those in Section 5.3.4 which established that core developers were most actively involved in task changes during the middle phases of their project, a time when they were also required to provide most knowledge to their teams.

Section 5.3.5 discussed the way core developers' attitudes were linked to their involvement in task changes. It was observed that when core developers were most work-focused they also made more changes. However, overall evidence in this work did not provide a conclusive link between core developers' expression of attitudes and their involvement in task performance. Thus, future work is encouraged to study this issue further.

Finally, Section 5.3.6 discussed the evidence of a link between core developers' involvement in task changes and their contribution of knowledge. In particular, these members asked more questions when they made fewer changes (and when less active practitioners were more involved). Additionally, core developers provided more answers and overall knowledge to their teams when they were actively involved in task performance. Given these findings, it is suggested that core developers should be surrounded by other capable communicators, and particularly when they are least active. Such a move would likely reduce the general negativity core developers expressed when they were least involved in their teams, and may help to maintain positive team climate.

Overall the results in this work show that the way practitioners used language is related to their performance and involvement in software development. These results confirm

that evidence of individuals' behaviours and attitudes is evident in their communications. The findings that were revealed in this work have implications for software engineering research, and particularly those findings aimed at providing understandings and retrospectives of the software development process. These implications are considered in the following chapter (Chapter 6).

5.4 Chapter Summary and Explanatory Model

This chapter discussed the results that were presented in Chapter 4 and highlighted the main findings of this work as are now shown in the explanatory model depicted in Figure 31 (which extends Figure 30 – note the green segments of the model). These findings were discussed in relation to previous theories, and while some previous patterns were confirmed, others were extended. Some patterns had not been observed previously and so provide the basis from which concrete theories may be derived. Section 5.1 surmised that distributed teams' communication patterns are linked to the approach that is utilised for software development, and regardless of the task type, team size, or duration of team tasks, only a few individuals dominate team interaction. It was observed that communication patterns established early were maintained throughout the software development project, and network density did not reflect accurately on participants' reachability (captured in the blue aspects of model in Figure 31). Additionally, it was noted that successful globally distributed agile software teams are highly connected, and this pattern is likely to mitigate for knowledge loss in centralised networks. Furthermore, Section 5.1 disclosed that practitioners who communicate heavily are also similarly involved in software development, and formal role assignment does not limit team members' performance once individuals are operating in a fluid organisational environment. Finally, evidence presented demonstrated that, regardless of the nature of the software tasks or team size, developers operating in all software roles are required to communicate, and particularly if such individuals were actively involved in software development (again noting the blue segments of the model in Figure 31).

Section 5.2 outlined that beyond actual involvement in software development, core developers exhibited attitudes that were ideal for maintaining team spirit and driving their teams' performance. These members were highly motivated and played a key role in their teams' outcome-oriented focus. Additionally, it was observed that successful distributed teams exhibit a range of behaviours, being both social and positive and task-

based. Such behaviours were acted out by various groups of practitioners and may have a balancing effect on team climate. Overall, Section 5.2 established that core developers occupied many informal roles (including informal leadership), and exhibited intrapersonal, interpersonal and organisation skills. Thus, formal role assignment may not predict communication and coordination requirements, and evidence shows that task involvement drives the need for communication and coordination (refer to the orange aspects of the model in Figure 31). Finally, it was shown that successful globally distributed agile software teams are highly task-focused, and that frequency-based analysis techniques do not capture fully the details of team dynamics during software development.

In the final section (Section 5.3) it was revealed that core developers expressed relatively consistent attitudes over the course of their project, but these members were most cognitive at project initiation and expressed some negative attitudes towards project closure. Core developers were most active in knowledge sharing and task performance during the middle phases of their project (refer to the green aspects of the model in Figure 31). Overall, the discussions above demonstrated that core developers' involvement in their teams was linked to their teams' demands, but these members also possessed cognitive abilities, and the more software development core developers did, the more they were required to provide knowledge and awareness to their wider teams. It was established that core developers' task performance is linked to their expression of work-related attitudes and their involvement in knowledge sharing (abstracted in green segments of the model in Figure 31). Finally, evidence in Section 5.3 confirmed that software practitioners' language processes are related to their involvement in knowledge sharing and task performance; refer to Figure 31 for abstractions.

These findings are evaluated in the following chapter (Chapter 6), and particularly, the implications of these findings for software project governance, software engineering theory and collaboration and process tools design are outlined.

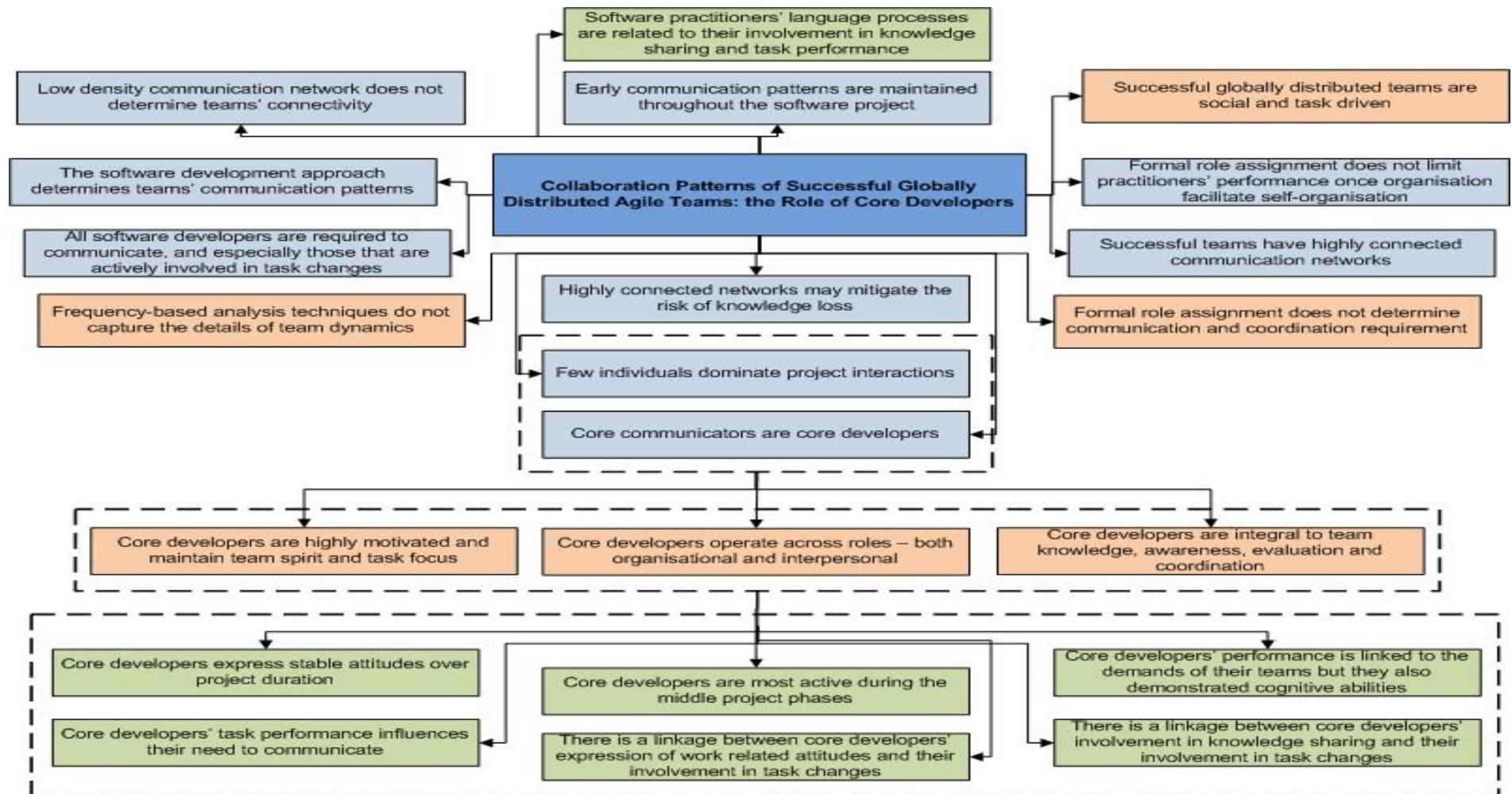


Figure 31. Collaboration patterns of successful globally distributed agile software teams, the true role of core developers and changes in core developers' attitudes, knowledge sharing and task performance

Chapter 6. Conclusions

The importance of studying the human factors relevant to software development has been emphasised throughout this thesis. Research in this space is posited to be particularly relevant for contemporary and future software engineering endeavours given inadequately performing software teams, even after a raft of recommendations in relation to software methods and tools. This research set out to study software human factors, and particularly those of globally distributed agile teams, largely through the comprehensive examination of their communication artefacts through multiple lenses. This chapter brings the research to a close, and provides the study's conclusions. First, Section 6.1 provides retrospections on the main findings that were revealed in this study. These retrospections form the basis for the other sections that follow. Section 6.2 outlines the novel research contributions provided by this work, considering how the work enhances theory, software engineering literature and pragmatic research in software engineering. This study is then evaluated and the work's limitations and threats are acknowledged in Section 6.3. The thesis then comes to a close with a summary of the implications for practice and research in Section 6.4.

6.1 Retrospections

This section provides a review of the results and discussions that were presented in Chapter 4 and Chapter 5. First, Section 6.1.1 summarises this study's outcomes for the enquiries around the collaboration patterns of successful globally distributed agile teams. This summary considers the findings that were aimed at answering research questions RQ1 – RQ 5. Section 6.1.2 then provides retrospections for the true role of core developers, largely considering the study outcomes for research questions RQ6 and RQ7. Finally, results and discussions that were aimed at answering RQ8 – RQ13, which considered task performance and its relationship to various characteristics and behaviours of core developers, are re-examined in Section 6.1.3.

6.1.1 Collaboration patterns (Phase 1)

In providing insights into the collaboration patterns of successful globally distributed agile teams, evidence in this work showed that IBM Rational Jazz teams communicated the most at project start-up and towards project completion (Licorish & MacDonell, 2013d). This finding is fitting for Jazz developments, and more generally for globally distributed developments where there are likely to be reduced opportunities for

communication and project oversight (Damian & Zowghi, 2003; Prikladnicki et al., 2003; Rudzki et al., 2010). An iterative approach to software development, where most requirements and features are elaborated in the middle project phases in line with the need for intensive team communication, may be less ideal for these settings (Prikladnicki et al., 2003; Tiwana, 2004). Globally distributed agile software development teams may lessen critical project overhead by reducing project uncertainties in the early project phase, thereby balancing agility with some levels of stable planning (Sharma & Kaulgud, 2011). Notwithstanding the specific way Jazz artefacts were normalised in this work, previous studies have also observed similar patterns for other global teams (Cataldo et al., 2006; Shihab et al., 2009; Yu et al., 2011). These findings are not universal however, as others have noted different communication patterns when studying other global teams (Datta et al., 2011), suggesting that other communication strategies may also facilitate globally distributed software development teams' success.

Another revealing finding of this research was the highly centralised network pattern noted for all of the ten teams that were studied, regardless of team size, task portfolio or number of iterations. This pattern has also been noted in other globally distributed settings (Crowston et al., 2006; Gacek & Arief, 2004; Mockus et al., 2002; Shihab et al., 2010), suggesting that not everyone needs to aggressively communicate for projects to succeed. While this pattern is understandable for OSS environments, where contributors' motivations are often divergent (Ljungberg, 2000; Markus et al., 2000; Oreg & Nov, 2008) and there are commonly pre-set hierarchical structures imposed on those involved (Crowston et al., 2006), the opposite is expected for commercial teams operating in globally distributed agile settings where practitioners are likely to benefit from comparable remuneration rewards. In these settings remuneration rewards may also impact practitioners' perceived moral obligations (Kankanhalli et al., 2005), and monitoring mechanisms in commercial environments also enforce communication (Grinter et al., 1999; Mockus et al., 2002) and other human resource strategies (Colomo-Palacios et al., 2010; Downey, 2009) that mandate active team involvement and highly skilled communicators. Thus, the evidence for the highly centralised Jazz teams' communication networks is somewhat revealing.

In fact, while the highly centralised pattern was observed for all ten Jazz teams, results in this work also show that all of the members of these teams were highly connected. This finding was also unexpected, particularly given previous beliefs around core

developers' important role in maintaining shared understandings (and team connections) (Crowston et al., 2006; Mockus et al., 2002). In fact, this position has also been long established in other disciplines (Bavelas, 1950; Leavitt, 1951). However, the results reported here show that even periphery members can remain highly connected and involved in knowledge sharing once teams establish strategic communication configurations. Periphery members were connected through their involvement on software tasks, whether or not core members were 'present'. This arrangement may be particularly useful for globally distributed agile software development teams, where distance naturally creates barriers to communication and there are likely to be few avenues for teams to maintain knowledge redundancy (Serce et al., 2009). Thus, an approach that limits redundant inter-connections among members, but at the same time ensures connectivity based on task assignment, is likely to reduce the overhead associated with managing multiple connections (James, 1951). This strategy could also serve as a cross training mechanism (Highsmith, 2000, 2004), and in the process, should mitigate loss in tacit knowledge should practitioners leave globally distributed agile teams (Boehm & Turner, 2003b; Williams & Kessler, 2003).

Findings regarding the communication networks' connectivity are not replicated in the task performance findings in this work. Additionally, in line with the centralised communication pattern noted, task changes were also heavily skewed for all ten Jazz teams (Licorish & MacDonell, 2013c). In fact, those practitioners that communicated heavily also maintained a similar level of presence in task (WI) changes. Although this pattern has been noted for other global teams (Cataldo & Herbsleb, 2008; Crowston et al., 2006; Robles et al., 2009), there was an associated expectation that highly active communicators would be project coordinators (Shihab et al., 2010), and so these members would not necessarily play active roles on software tasks. However, findings in this research contradict this conjecture, and show that Jazz core communicators are also core developers (Cataldo & Herbsleb, 2008; Shihab et al., 2010). Accordingly, task involvement seems to impact the need or tendency to communicate (Cataldo & Herbsleb, 2008).

In fact, rather than being project coordinators the Jazz core developers occupied various formal roles, and most were programmers (Licorish & MacDonell, 2013c). This finding also endorses the viewpoint that active involvement in software tasks is likely to drive practitioners' need to communicate. Thus, highly active programmers are likely to benefit from good communication, teamwork and social skills; a finding that is contrary

to previous belief (Acuna et al., 2006; Andre et al., 2011). Findings indicating Jazz programmers' high level of involvement in team communication are particularly surprising given that each Jazz team was led by a formal leader - more than that, all of the teams studied comprised multiple members occupying both project manager and team lead roles. This finding indeed suggests that task involvement may impact the need to communicate (Cataldo & Herbsleb, 2008), as against being driven by practitioners' role assignment. This unexpected pattern of involvement in communication for Jazz practitioners occupying the formal programmer role has also been observed for teams fixing bugs (Datta et al., 2010). Those operating outside their formal role are said to be integral to their team's self-organisation process (Hoda et al., 2010b), a necessary requirement for mitigating the impacts of geographical and temporal distances in globally distributed development contexts (Espinosa et al., 2006). Those behaviours, and particularly the evidence noted for core developers' actual involvement in both communication and task performance, also signal that these members are likely to be driven intrinsically (Chang et al., 2013; Constant et al., 1996). However, a supportive organisation culture may be necessary for encouraging these members' performance, and for enabling practitioners' output based on their work demands (Powell, 1990), particularly for globally distributed agile teams.

The retrospections just presented pertain to RQ1 – RQ5 (refer to Section 2.6). These questions were answered through the preliminary data analysis considered in Section 4.1, which uncovered that Jazz teams communicated most at project start and completion, a few individuals contributed most of the teams' communication, Jazz teams' communication networks were highly connected through task assignment, core communicators were also core developers, and formal role assignment did not limit practitioners' contributions. While these preliminary findings are insightful, several of these patterns were also previously noted for other globally distributed teams. That said, although these findings were noted, and particularly those of the centralised communication structure and the skewness in practitioners' task performance, previous work did not examine in detail the true role of these core developers. This is despite core developers' critical role in their teams' knowledge sharing processes and task performance, and the potential for such knowledge to inform team composition strategies (refer to Section 2.7.1). In contrast, this research addressed this gap, retrospections of which are provided in the following section (Section 6.1.2).

6.1.2 The true role of core developers (Phase 2)

While it has been previously discovered that a few developers in a team dominate project communications and that these individuals make most of the task changes during their teams' software projects (Cataldo & Herbsleb, 2008; Crowston et al., 2006; Gacek & Arief, 2004; Mockus et al., 2002; Shihab et al., 2010), the rationale for this phenomenon has not been provided. Details around the reasons for these practitioners' distinct presence and performance, and insights into how these members contribute to team dynamics, have not been revealed – understandings that are likely to be beneficial in terms of informing team composition strategies. This study applied psycholinguistic and directed CA techniques to the study of artefacts from globally distributed agile Jazz teams to address these gaps.

Findings in this work show that core developers expressed lower amounts of self-focused attitudes, and these members used more delegation language than their counterparts. These findings are revealing given core developers' heavy involvement in undertaking software tasks. However, this evidence is also positive for Jazz teams given core developers' central position in their teams' communication and their likely impact on overall team climate (Licorish & MacDonell, 2013c), and particularly for globally distributed developments where a positive team climate and shared team norms are necessary to cultivate team trust (Dullemond et al., 2009; Lee & Yong, 2010). Those that exhibit individualistic attitudes have been shown to negatively affect team synergies and performance (Benne & Sheats, 1948). In particular, had core developers exhibited these traits, their teams would likely be affected negatively, as team norms are generally cultivated by the leaders' actions (Chang et al., 2013; Denning, 2012). Thus, the expression of self-focused behaviours by the most influential team members is likely to impact agile global teams' behavioural climate negatively, especially given reduced incidence of face-to-face communication (Chang & Ehrlich, 2007; Espinosa et al., 2006), and thus, team trust (Al-Ani et al., 2011; Krebs et al., 2006; Zigurs, 2003). Evidence in this work of high levels of delegation language use by Jazz core developers suggests that these practitioners assumed leadership status in their teams. This finding is insightful when considering that core developers comprised only 15 of the 146 contributors in the ten teams, and fewer than 50% of these members were formal team leaders.

Apart from delegation, Jazz core developers also exhibited strong task focus. Role theories have shown that individuals that exhibit this trait are often keen on task performance and consequently they drive their teams towards achieving project targets (Belbin, 2002; Benne & Sheats, 1948). These behavioural processes are also associated with ambition and commitment (Chang et al., 2013; Denning, 2012). Given Jazz core developers' expressions of these desirable traits, and their central position in their teams, these Jazz teams were likely to cultivate achievement-driven team norms (Denning, 2012). Such a team atmosphere may be ideal for globally distributed agile environments. Of particular significance is the previous finding that experience and education did not account for core developers' demeanour (Cataldo et al., 2006; Curtis et al., 1988), thus, Jazz core developers' commitment to their teams' performance is likely to be driven by more intrinsic motivations (Chang et al., 2013).

Jazz's less active contributors expressed cognitive attitudes, and exhibited collective focus. Cognitive attitudes have been linked previously to task performance (Andre et al., 2011), whereas, collective processes are linked to team focus and shared team norms (Denning, 2012; Tuckman, 1965). These behaviours are desirable for teams' performance, and may be particularly beneficial for globally distributed agile teams. Overall, Jazz teams demonstrated a mix of both social and task-focused attitudes. These traits are said to have a balancing effect during team work (Benne & Sheats, 1948). Those exhibiting social traits contribute towards positive group climate, promoting harmonizing and compromising team norms, while task-driven behaviours are generally associated with task success, contributing and initiating ideas and knowledge towards task completion. These behavioural processes have a positive influence on team morale and task outcomes (Denning, 2012).

Results in this research have further revealed that core developers engaged heavily in vertical communication, and contributed expression related to team organisation and task assignment (Licorish & MacDonell, 2013c). Such forms of communication are associated with those in power and so are normally befitting of those in leadership positions or those assigned to coordination roles (Andre et al., 2011). Core developers also acted as information hubs and offered their teams a diverse range of solutions and advice. These practitioners' high levels of message contribution made them very aware of their project landscape which showed in the number of questions they answered and the volume of information they contributed (Kanawattanachai & Yoo, 2007; Palazzolo et al., 2006). Additionally, in line with the task involvement assessment noted in Section

6.1.1, evidence derived from the contextual analysis conducted in this work also shows that core developers occupied problem solving and skill based roles; that is, roles concerned with the practical translation and application of concepts and plans developed by the team and putting forward ideas and strategies for achieving the objectives adopted by the team (Belbin, 2002). Furthermore, in support of the linguistic analysis results indicating low levels of individualistic attitudes, contextual analysis results show that core developers were interpersonal (Downey, 2009).

The presence of core developers was no doubt critical for their globally distributed agile teams to maintain outcome-oriented and shared team norms. That said, core developers also exhibited some judgemental attitudes, behaviours that are positive for enhancing innovativeness and critical evaluation among group members (Tjosvold, 2008); however, when excessive, such behaviours may become counterproductive (Belbin, 2002; Benne & Sheats, 1948), and so further analysis (reviewed in Section 6.1.3) to understand how and when these behaviours become pronounced was appropriate.

Overall, as a group, Jazz developers (core and others) spent most of their time engaging one another in regard to software tasks. This keen interest in task outcomes is pleasing given that these teams were separated by distance. These findings also indeed suggest that core developers positively influenced their teams' behavioural norms in that these members were highly task focused, and a task-driven outlook may be a positive sign for team performance. That said, however, the high intensity of task-focused communication suggests that Jazz teams would benefit from tools that enable communication prioritisation. In fact, such a capability may generally support globally distributed agile software teams, and particularly those that need to maintain task focus.

The retrospections above were aimed at providing reflections regarding RQ6 and RQ7 (refer to Section 2.7.1). These questions were answered through the overall project snapshot analyses, comprising linguistic analysis and directed content analysis results as presented in Section 4.2. Evidence in this work discovered that Jazz core developers exhibited behaviours that are integral for maintaining collective project visions, task performance and team dynamics. It was also revealed that core developers occupied various informal roles in their teams, and they were central to their teams' actual task coordination and knowledge sharing. Further, it was observed that, as a group, Jazz developers spent most of their communication effort engaging about software tasks. Contextual findings in this work supported the preliminary proposition that Jazz core

developers' performance is directly related to an organizational environment that promotes informal and organic work structures. This form of organization configuration may be necessary for agile teams, and especially for distributed developments.

While core developers were no doubt invaluable to their teams' performance, these members were also seen to exhibit some judgemental attitudes. When contributed in large amounts, these behaviours may be undesirable for task performance. Additionally, since there may be some initial team arrangements that caused core developers to become hubs in their teams, knowledge of what motivated core developers could be invaluable in coordinating efforts aimed at detecting and moulding such 'software gems'. The next stage of the analysis therefore investigated how core developers' attitudes and knowledge sharing behaviours change over time and the relationship between attitudes, knowledge sharing behaviours and task performance. These retrospections are provided in the following section (Section 6.1.3).

6.1.3 Changes in core developers' attitudes, knowledge sharing and task performance (Phase 3)

In extending the reviews above (refer to Section 6.1.1 and Section 6.1.2) this section considers the retrospections regarding changes in core developers' attitudes, knowledge sharing behaviours and task performance, and the way these are related. Findings for the way core developers expressed attitudes over the course of their project show that core developers exhibited relatively stable feelings during their project, but expressed most desirable team attitudes at project inception and were most cynical towards project completion. While software practitioners were previously found to exhibit less optimism as their project progressed (Rigby & Hassan, 2007), suggesting that core developers' less desirable attitudes at project closure may have been related to release pressures or increasing levels of task difficulty (Cataldo & Herbsleb, 2008), evidence in this work also revealed that these members were less engaged in communication and task changes when they exhibited their dissatisfaction. Taken together, these findings suggest that core developers were potentially frustrated at their reduced involvement and likely discontented with the feedback they received from the wider team during their less active periods. Additionally, of particular note is the evidence that core developers also exhibited their highest level of task-focus in their communications when they were least involved with their teams. Such a finding confirmed that these members were highly motivated, and were keen on driving their teams forward (Allen & Meyer, 1990; Morgan & Hunt, 1994). While the negative attitudes expressed by core

developers towards project completion are unhelpful for team climate (Benne & Sheats, 1948; Goldberg, 1981), and particularly given the central position these practitioners occupied in their teams' communications (Licorish & MacDonell, 2013c), core developers also expressed high levels of social and positive attitudes throughout their project, and more so at project start-up. Such attitudes are critical to team formation and establishing team dynamics (Downey, 2009; Tuckman, 1965), which may mitigate the impact of later negativity and team issues. Additionally, given that core developers were least involved in their teams' communication towards project completion, their negative behaviours at this time perhaps had the least harmful impact on their teams' behavioural norms. That said, however, negative behaviours are not ideal for globally distributed agile software developments given the constraints introduced by distance in these setting (Chang & Ehrlich, 2007; Espinosa et al., 2006), and so, wider team participation should be encouraged during core developers' least active periods to mitigate the likely negative effects of these members' unhappiness.

Jazz core developers shared most of their knowledge during the middle project phases, and these members were most active in contributing to their teams' knowledge pools when they were performing task (feature) changes. Jazz core developers also asked questions primarily at project start and project completion. While a large number of questions at project start is understandable given the need for project scoping and requirements clarification at this time, core developers' high level of questions at project completion coincided with their reduced presence. This evidence supported the assessment that these members' expression of negative attitudes was related to their limited involvement at project completion, and their desire to maintain team performance (cf. core developers' task focus attitudes at project completion) (Kline & Peters, 1991; Mowday et al., 1979). Results in this work also demonstrated that core developers' expressions of judgemental behaviours were aligned with their active involvement in task performance, suggesting that critical evaluations may aid in task performance (Tjosvold, 2008) during globally distributed agile projects. These critical evaluations are likely to result in innovative solutions.

In terms of the initial team arrangements that influenced core developers' distinct performance in their teams, results in this work revealed that these members exhibited cognitive competencies at project initiation; however, the position these practitioners occupied in their teams was linked to their active involvement in task changes. This involvement drove the need for core developers' distinct presence in their teams'

knowledge sharing networks. In particular, these members were most active in their teams' communication when they also made most task changes, tending to also share most knowledge and contribute most positively to team climate. The opposite is noted when core developers were least involved. This finding has implications for globally distributed agile software project governance, and particularly for the management of software teams' tacit knowledge (considered later in this chapter).

In fact, although this work did not provide a conclusive link between core developers' expression of attitudes and their involvement in task changes, results show that core developers contributed most changes when they used more work-related terms. These results provide some support for the linkage between attitudes and team performance, in support of social motivation theories (Brock & Kim, 2002; Geen, 1991; Inkpen & Tsang, 2005; Levin & Cross, 2004; Quigley et al., 2007). More conclusive findings were drawn from this research in relation to core developers' knowledge sharing behaviours and their involvement in task performance. For instance, evidence presented here showed that when there were higher levels of questioning from core developers these practitioners were less active in task changes, tending to rely more on their wider teams. This evidence supported the conjecture above that core developers were unhappy due to their lack of awareness. Perhaps they were indeed dissatisfied with the feedback they received at this time (Goldberg, 1981). In contrast, when core developers were actively involved in task changes they provided more answers, information, ideas, and instructions.

This section has provided retrospections around the findings related to RQ8 – RQ13. Linguistic analysis and directed content analysis techniques were used to study core developers' artefacts using a longitudinal approach to address these questions. Findings from these analyses indicated that core developers contributed relatively stable attitudes over the course of their project, these members were most active in knowledge sharing during the middle phase of their project and their attitudes and involvement in knowledge sharing were linked to the demands of their wider teams. These practitioners also brought high levels of skills and cognitive characteristics to their teams. These individuals started their project providing high levels of ideas, suggestions, information, comments, instructions and answers, and quickly became the centre of their teams' knowledge activities as the Jazz project progressed. These patterns were related to core developers' actual involvement in task (WI) changes – the more changes core developers performed the more knowledge they provided. When these practitioners

were least involved in communication and task changes, they asked more questions and exhibited negative team attitudes, suggesting that they were dissatisfied with the feedback they received from the wider team. These findings have implications for future research, software project governance and collaboration and process tool enhancements. These issues are considered in Section 6.4. Prior to their consideration, the following section (Section 6.2) outlines this research's contributions, and Section 6.3 acknowledges this study's limitations and threats.

6.2 Research Contributions

This section outlines the novel contributions that have been made as a result of this research. In terms of the contributions to theory, this research provides conjecture that may form the basis of initial explanation theories, comprising multiple aspects related to the collaboration patterns of successful globally distributed agile teams and the way core developers contribute to their teams' dynamics. These contributions are outlined in Section 6.2.1. The contributions that this work provides to the software engineering literature are summarised in Section 6.2.2, primarily comprising extensions to the literature around core developers' contributions to team dynamics. Section 6.2.3 finally summarises this work's contributions to pragmatic research, with some particular suggestions for those studying software repository data.

6.2.1 Contributions to Theory

Theories take different shapes and forms depending on the paradigm under consideration. While the process of theorising is well established in other disciplines (e.g., natural sciences (Kuhn, 1970) and psychology and social sciences (Davis, 1971; Dubin, 1978)), there is less focus on this issue in software engineering (Hannay et al., 2007). Thus, some researchers have argued that this lack of appreciation for the theory generation process has often led to poor research outcomes, and the immaturity of the software engineering discipline (Johnson et al., 2012). In contrast, research practitioners in the IS discipline tend to place more emphasis on the theorising process (Hannay et al., 2007; Morrison & George, 1995), and have also provided frameworks for theorising that can encompass software engineering research (Gregor, 2006). These frameworks were used to inform the theoretical basis that guided the generation of this study's research questions (refer to Chapter 2), and the theory formulation process (refer to Section 3.5).

Of the various theoretical representations that are presented by Gregor (2006), this work contributes initial conjectures that may form the basis of explanation theories (in addition to confirmations). Theories that are generated by empirical software engineering research typically take this form (Hannay et al., 2007), and generally provide understandings of the phenomena under consideration (Gregor, 2006). The conjectures that may form future explanation theories in this work are aligned with the multistage analysis approach adopted in this case study, comprising both confirmatory and exploratory aspects (refer to Section 3.4).

Theoretical contributions from the confirmatory analysis relate to the understanding of the collaboration patterns of successful globally distributed agile teams, as aligned with the confirmatory analysis that was performed in order to replicate previous findings (refer to Section 4.1), and inform deeper exploratory analysis. Evidence from the confirmatory analysis shows that the software development approach determines the team's communication patterns, few individuals dominate project interaction, low density communication networks do not determine teams' connectivity, early communication patterns are maintained throughout the software project, all software developers are required to communicate, especially those that are actively involved in task changes, successful teams have highly connected communication networks, core communicators are core developers, and formal role assignment does not limit practitioners' performance once the organisation facilitates self-organisation. These outputs are summarised in Table 21, where the status of each proposition is also highlighted.

Exploratory aspects of the conjectures that are contributed in this work relate to how and why core developers contribute to globally distributed agile team dynamics. This objective was aligned to the second and third phases of the data analysis conducted during this research. In the second phase core developers' attitudes and their enacted roles were studied from a project snapshot perspective (refer to Section 4.2). Evidence discovered during this phase provides multiple insights to the initial conjectures that may form explanation theories that are provided here (refer to Table 21). It was revealed that successful globally distributed teams are social and task driven, formal role assignment does not determine communication and coordination requirements, core developers are highly motivated and maintain team spirit, core developers operate across roles – both organisational and interpersonal, and core developers are integral to team knowledge, awareness, evaluation and coordination. Additionally, evidence

revealed in the second phase of the data analysis confirmed that frequency-based analysis techniques do not capture the details evident in team dynamics.

Table 21. Summary of theoretical contributions

Study Area	Proposition or Conjecture	Status
Collaboration patterns of successful globally distributed agile software teams	The software development approach determines teams' communication patterns.	Confirmation
	Early communication patterns are maintained throughout the software project.	New Conjecture
	Successful teams have highly connected communication networks.	New Conjecture
	Low density communication network does not determine teams' connectivity.	New Conjecture
	Few individuals dominate project interaction.	Confirmation
	Core communicators are core developers.	Confirmation
	All software developers are required to communicate, and especially those that are actively involved in task changes.	New Conjecture
	Highly connected networks may mitigate the risk of knowledge loss.	New Conjecture
	Formal role assignment does not limit practitioners' performance once the organisation facilitates self-organisation.	Confirmation
	Formal role assignment does not determine communication and coordination requirement.	New Conjecture
	Successful globally distributed teams are social and task driven.	New Conjecture
	Software practitioners' language processes are related to their involvement in knowledge sharing and task performance.	New Conjecture
The true role of core developers	Core developers are highly motivated and maintain team spirit and task focus.	New Conjecture
	Core developers operate across roles – both organisational and interpersonal.	New Conjecture
	Core developers are integral to team knowledge, awareness, evaluation and coordination.	New Conjecture
Changes in core developers' attitudes, knowledge sharing and task performance	Core developers express stable attitudes over project duration.	New Conjecture
	Core developers are most active during the middle project phases.	New Conjecture
	Core developers' performance is linked to the demands of their teams but they also demonstrated cognitive abilities.	New Conjecture
	Core developers' task performance influences their need to communicate.	New Conjecture
	There is a linkage between core developers' expression of work related attitudes and their involvement in task changes.	New Conjecture
	There is a linkage between core developers' involvement in knowledge sharing and their involvement in task changes.	New Conjecture
Analysis techniques for repository artefacts	Frequency-based analysis techniques do not capture the details of team dynamics.	Confirmation

The final aspect of this work's theoretical contribution relate to the changes in core developers' attitudes, knowledge sharing and task performance, and how these elements were associated. These findings were derived through longitudinal analyses of core developers' artefacts (refer to Section 4.3). Evidence from these analyses revealed that core developers express relatively stable attitudes over their project duration, core developers are most active during the middle project phases, core developers' task performance influences their need to communicate, core developers' performance is linked to the demands of their teams but they also demonstrate cognitive abilities, and core developers' language processes are related to their involvement in knowledge sharing and task performance. These outputs are also provided in Table 21.

While findings from isolated case studies may not adequately deliver theories as such, such an approach provides early conjectures from which theories may be derived from future empirical work (Potts, 1993; Runeson & Host, 2009). Accordingly, the consolidated model in Figure 31 and the detailed summary in Table 21 represent propositions; further research should therefore replicate this work to extend these findings (refer to Section 6.4.2). Such work would also more generally contribute to the software engineering knowledge base for globally distributed agile software developments. The specific contributions provided by this work are reviewed next (in Section 6.2.2).

6.2.2 Contributions to SE Literature

This work provides multiple contributions to the body of software engineering literature around globally distributed agile team dynamics. First, this work follows previous studies (Cataldo et al., 2006; Datta et al., 2011; Shihab et al., 2009; Yu et al., 2011), and provides insights into the collaboration patterns of successful globally distributed agile teams. Specifically, this study has replicated other works focusing on changes in teams' communication patterns (Cataldo et al., 2006; Datta et al., 2011; Shihab et al., 2009; Yu et al., 2011), and has provided insights into teams communication dynamics. Evidence from this aspect of the work were largely confirmatory. In particular, this work discovered centralised communication patterns for the teams that were examined, thus, demonstrating convergence with other works in the globally distributed development space (Cataldo et al., 2006; Shihab et al., 2009; Yu et al., 2011). However, while the centralised pattern is noted in the findings of this work, it was also observed that the globally distributed teams that were studied were highly connected through their task involvement. This pattern had not been observed previously, and so there is need for follow up research to ascertain the effectiveness of such a pattern for team knowledge distribution. This work contributes to the software engineering literature on the way highly active communicators are involved with task performance, and again, the evidence in this work converges with those of previous studies which revealed that core communicators were actually core developers (Cataldo & Herbsleb, 2008; Crowston et al., 2006; Robles et al., 2009). This evidence was particular surprising given that the teams studied in this work operated in a commercial rather than open source context. Furthermore, this work's assessment of the roles of core developers contributes understandings to the software engineering literature around self-organising roles and work structures (Hoda et al., 2010b).

In terms of the reason for the centralised pattern noted for globally distributed agile software development teams, this work has provided understandings through the use of psycholinguistic (Pennebaker et al., 2007; Pennebaker & King, 1999) and contextual analysis methods (Henri & Kaye, 1992; Zhu, 1996) regarding the true role of these core developers (Licorish & MacDonell, 2013c). These explanations represent an extension of the literature that previously reported this pattern, and particularly those works that have largely employed frequency-based analysis approaches (Cataldo et al., 2006; Crowston et al., 2006; Shihab et al., 2009; Yu et al., 2011).

Although this study examined multiple teams of one case organisation, and so the findings presented therein may not generalise to all software teams (Runeson & Host, 2009) (refer to Section 6.3 for details), this work contributes to the literature around attitudes and behaviours that are exhibited by successful globally distributed agile software development teams (Licorish & MacDonell, 2013c). Novel insights into the attitudes and behaviours of core developers are provided, along with conjectures about these members' motivation (Chang et al., 2013; Denning, 2012). Further, this work provides insights into the roles that are enacted by core developers, and members of globally distributed agile teams in general. While such evidence have been provided for software teams along the lines of self-organising principles (Hoda et al., 2010b), this research represents one of the first attempts to understand the inner details of globally distributed agile software development teams (Rigby & Hassan (2007) is the only other study that has used linguistic analysis techniques, one of the approaches that was used in this work), and particularly, insights into the true role of core developers from the study of their artefacts.

Additionally, having established core developers' true role in their teams, this study explored the way these members' attitudes and knowledge behaviours change as their project progresses, the specific team arrangements that are responsible for core developers' distinct performance in their teams, how these members are involved in task (WI) changes over the course of their project and how their behaviours are related to their involvement in task changes. These understandings provide an extension of the findings that were introduced in the second phase of the data analysis around core developers' true roles (refer to Section 6.1.3 for retrospections) (Licorish & MacDonell, 2013c). While there have been prior efforts aimed at studying the way teams evolve (Capiluppi et al., 2007; Rowley & Lange, 2007), enquiries in this research represent the first attempt to examine how core developers contribute to their teams' process over the

course of their project (through the use of deeper text analysis approaches), and how these members' organizational, interpersonal and intrapersonal competencies sustain their project's health.

Furthermore, this work has provided insights into commercial globally distributed agile software teams, an enhancement to the more frequently used OSS repositories regularly extracted to study process issues (Bachmann & Bernstein, 2009; Bird et al., 2006a). As stated earlier (refer to Section 3.4.1 for details), the challenges with using OSS repositories are related to the reliability and validity of the data available. Research evidence has reported poor data quality in repositories of OSS projects (Aune et al., 2008; Bird et al., 2006a). While data pre-processing techniques are generally used for improving data quality, concerns remain over the reliability of data in such sources (Rodriguez et al., 2012). This work was not faced with such issues, and so, the findings that are provided in this study are likely to be highly reliable (refer to Section 6.3 for reliability discussions).

Of final note is the contribution provided to the software engineering literature through the utilisation of contextual analysis techniques to understand team processes. As noted earlier (and throughout Chapter 2), studies in the software engineering discipline examining human factors from communication artefacts and repository data have tended to employ analytical and frequency-based approaches (Abreu & Premraj, 2009; Bird et al., 2006a). Such approaches align with a technical focus on team processes (Glass et al., 2002). Although certainly useful, it is generally understood (and there is growing recognition in SE) that these approaches would benefit from triangulation with more contextual techniques (Di Penta, 2012; Johnson & Onwuegbuzie, 2004). This view is also supported by researchers working in the IS discipline (Klein & Myers, 1999), where tested research approaches in the behavioural sciences, management and psychology domains have been recommended for use when studying human aspects of IS processes (Vessey et al., 2002). Additionally, previous work has placed significant emphasis on the need for longitudinal studies to understand changes in teams' activities over time, in addition to the frequently used snapshot or cross-sectional analysis approaches (Cataldo & Ehrlich, 2012; Hinds & McGrath, 2006). This work's utilization of deeper contextual analysis techniques (Licorish & MacDonell, 2013b), and the incremental systematic approach used for studying Jazz teams artefacts (refer to Section 3.4 for details), from both static and temporal perspectives, also represent contributions to software engineering literature (Licorish & MacDonell, 2013a). In particular, the

approach to embrace the general principles of pragmatism during the examination of software teams' artefacts is novel (Licorish & MacDonell, 2013a), and in itself represents a contribution to the software engineering knowledge base. This issue is considered further in the following section (Section 6.2.3).

6.2.3 Contribution to Pragmatic Research in SE

Analytical approaches may not adequately reveal the details in human interactions (Runeson & Host, 2009). While this viewpoint has been expressed repeatedly by those researching the human factors involved in software development (Easterbrook et al., 2008; Glass et al., 2002; Goguen, 1993; Klein & Myers, 1999; Ramesh et al., 2004; Yu et al., 2011), software engineering researchers, and particularly those that explore team processes from artefacts and repository data, have tended to focus mainly on mathematically-based analysis techniques (Bird et al., 2006a; Cataldo & Ehrlich, 2012; Datta et al., 2010; Ehrlich & Cataldo, 2012; Hinds & McGrath, 2006; M. Zhou & Mockus, 2011). This represents a limitation to the evidence these works are able to provide, and so, more flexible design strategies are generally encouraged to limits such threats (Leech & Onwuegbuzie, 2009; Onwuegbuzie & Leech, 2005; Schultz & Hatch, 1996).

This research utilised such a flexible approach, comprising confirmatory and exploratory techniques under a case study design, and so contributes insights for those intending to adopt a pragmatic approach in the study of software repository data (Licorish & MacDonell, 2013a). Combining these techniques in this way helped this work to uncover multiple perspectives around team dynamics, and so demonstrate the utility of pragmatism for software engineering research. The employment of such a pragmatic approach also provided multiple avenues for triangulation, including both methodological and observer triangulation (Runeson & Host, 2009; Stake, 1995). This is particularly necessary for investigating how software development is conducted by individuals, whose behaviours are often nuanced in complex social systems (Goguen, 1993), and particularly when using artefacts (Di Penta, 2012; Robles et al., 2009).

For instance, data pre-processing techniques are generally recommended for maintaining data quality before data mining software are applied to the given dataset. The use of these techniques is said to enhance the validity of study results (Larose, 2005; Rodriguez et al., 2012). While this work did not perform data mining to make predictions or classifications as such, data pre-processing techniques were utilised to

explore and extract the Jazz repository. Such techniques are typically used under a positivist framework, and normally lead to prediction and classification. However, employing data pre-processing techniques to enhance data quality may not be confined to studies that are aimed at prediction and classification, and such techniques provide general utility for research best practice. Output from the data pre-processing stage in this research informed the sampling process and also provided preliminary insights into the way successful globally distributed agile teams collaborate, providing the platform for multiple forms of methodological triangulation.

SNA techniques were subsequently employed to study teams' interactions, providing an increment to the methodological triangulation process. There is growing recognition that exclusively utilising SNA techniques to study interactions restricts the level of inferences researchers are able to make (Easterbrook et al., 2008; Erlin et al., 2008; Jamali & Abolhassani, 2006). Accordingly, deeper approaches were used to supplement SNA techniques in this work (refer to Table 21 for conjecture). However, prior to the employment of these deeper techniques this work employed an exploratory approach (in addition to quantitative measures and statistical analysis techniques) to the study of team sociograms by qualitatively examining these graphs. Evidence revealed through these qualitative examinations converged with those drawn from the quantitative measures, offering this work a further tier of methodological triangulation.

Another layer of methodological triangulation was provided through the use of linguistic analysis techniques in this work. Application of such a richer textual analysis technique provided multiple forms of insights that the SNA-related measures did not previously reveal. Evidence revealed though the use of this technique extended those provided in previous work and provided further understandings into globally distributed agile software teams' dynamics.

Content analysis was then used to overcome the limitations of the linguistic analysis techniques, and this form of data analysis extended the triangulation efforts, in terms of both methodological and observer triangulation. Further, the linguistic analysis results were correlated with the directed CA results in a form of experimental assessment, to support methodological triangulation efforts. These techniques were combined to provide multiple insights, and the usage of these techniques support the utility of mixed method research, and particularly for those considering the study of repository data.

In closing this section, no single technique (SNA, linguistic analysis or directed CA) would reveal the research findings that were revealed in this work through the use of a mixed method approach. Additionally, the use of a pragmatic approach in this research reduced the threats associated with exclusively adopting either a confirmatory or exploratory research design. That said however, this research still suffers from some shortcomings, and particularly those that typically accompany case studies. This issue is considered further in the next section (Section 6.3).

6.3 Research Evaluation, Limitations and Threats

Although the findings in this work are novel, this study, like any other case study (Runeson & Host, 2009), suffers from limitations that may present threats to the work's generalisability. Runeson & Host (2009) suggest that the best assessment for the quality of case studies is a comprehensive evaluation of the study design, data collection, analysis of the collected data and the reporting of study findings. Similarly, Yin (2003) suggests that the findings of case studies are meaningful when they are presented with an assessment of construct validity, internal validity, external validity and reliability. Perry, Sim, & Easterbrook (2004) recommended that case studies should outline research questions very early in the study, data collection should exhibit consistency, inferences should be made from the data in answering research questions, the study should provide an explanation of the phenomenon, and the study threats and validity should be addressed in a systematic way.

These processes are all captured in Runeson & Host (2009) guidelines, and are considered in this work under the research evaluation taxonomy provided in Table 22. In Table 22, Runeson & Host (2009) checklist items are provided in the first column, and the second column provides pointers to the specifics steps that were taken in this work to address the corresponding issue. This measure is combined with Yin (2003)'s tests for evaluating the quality of case studies which are subsequently provided.

Table 22. Research evaluation taxonomy

Checklist Item	Addressed in this study?
Case study design	
1. What is the case and its units of analysis?	Refer to Section 3.4
2. Are clear objectives, preliminary research questions, hypotheses (if any) defined in advance?	Yes (Refer to Section 1.3, Section 1.5, Section 2.6, Section 2.7.1, Section 2.7.2 and Section 2.7.3)
3. Is the theoretical basis - relation to existing literature or other cases - defined?	Yes (Refer to Chapter 2)
4. Are the authors' intentions with the research made clear?	Yes (Refer to Section 1.1, Section 1.2, Section 1.3, Section 1.5, Chapter 2 and Chapter 3)

Checklist Item	Addressed in this study?
5. Is the case adequately defined (size, domain, process, subjects...)?	Yes (Refer to Section 3.4)
6. Is a cause - effect relation under study? If yes, is it possible to distinguish the cause from other factors using the proposed design?	No (Refer to Section 2.6, Section 2.7.1, Section 2.7.2 and Section 2.7.3)
7. Does the design involve data from multiple sources (data triangulation), using multiple methods (method triangulation)?	Study used multiple forms of data from a representative repository, multiple methods were used for triangulation (Refer to Section 3.4)
8. Is there a rationale behind the selection of subjects, roles, artefacts, viewpoints, etc.?	Yes (Refer to Section 3.4.1, Section 3.4.2, Section 3.4.3.1, Section 4.1.2 and Section 4.1.3)
9. Is the specified case relevant to validly address the research questions (construct validity)?	Yes (Refer to Section 3.4.1, Section 3.4.2, Section 3.4.3.1)
10. Is the integrity of individuals/organizations taken into account?	Yes (Refer to Appendix X (Confidentiality Agreement))
Preparation for data collection	
11. Is a case study protocol for data collection and analysis derived (what, why, how, when)? Are procedures for its update defined?	Study design used as protocol, procedures were updated throughout this research project (Refer to Section 3.4)
12. Are multiple data sources and collection methods planned (triangulation)?	Multiple forms of data gathering planned and examined from a representative repository (Refer to Section 3.4.2)
13. Are measurement instruments and procedures well defined (measurement definitions, interview questions)?	Yes (Refer to Section 3.4.3, Section 3.4.4 and Section 3.4.5)
14. Are the planned methods and measurements sufficient to fulfil the objective of the study?	Yes (Refer to Section 1.3, Section 3.4.2, Section 3.4.3, Section 3.4.4 and Section 3.4.5)
15. Is the study design approved by a review board, and has informed consent obtained from individuals and organizations?	This research project was exempted from AUT's formal ethics review process given the study design and completion of IBM's Confidentiality Agreement (Refer to Section 3.4.5.8 and Appendix X (Confidentiality Agreement))
Collecting Evidence	
16. Is data collected according to the case study protocol?	Yes (Refer to Section 3.4.1 and Section 3.4.2)
17. Is the observed phenomenon correctly implemented (e.g. to what extent is a design method under study actually used)?	Yes (Refer to Section 3.4.3, Section 3.4.4, Section 3.4.5 and Chapter 4)
18. Is data recorded to enable further analysis?	Yes (Refer to Section 3.4.2, Section 3.4.3.1, Section 3.4.4.2 and Section 3.4.5.8)
19. Are sensitive results identified (for individuals, the organization or the project)?	No specific sensitive results identified (Refer to Section 3.4.5.8 and Appendix X (Confidentiality Agreement))
20. Are the data collection procedures well traceable?	Yes (Refer to Section 3.4)
21. Does the collected data provide ability to address the research question?	Yes (Refer to Chapter 4 and Chapter 5)
Analysis of collected data	
22. Is the analysis methodology defined, including roles and review procedures?	Yes (Refer to Section 3.4.3, Section 3.4.4, Section 3.4.5 and Chapter 4)
23. Is a chain of evidence shown with traceable inferences from data to research questions and existing theory?	Yes (Refer to Chapter 4 and Chapter 5)
24. Are alternative perspectives and explanations used in the analysis?	Yes (Refer to Chapter 5)
25. Is a cause - effect relation under study? If yes, is it possible to distinguish the cause from other factors in the analysis?	No (Refer to Section 2.6, Section 2.7.1, Section 2.7.2 and Section 2.7.3)
26. Are there clear conclusions from the analysis, including recommendations for practice/further research?	Yes (Refer to Chapter 6)
27. Are threats to the validity analysed in a systematic way and countermeasures taken? (Construct, internal, external, reliability)	Yes (Refer to Section 6.3)
Reporting	
28. Are the case and its units of analysis adequately presented?	Yes (Refer to Section 3.4 and Chapter 5)
29. Are the objective, the research questions and corresponding answers reported?	Yes (Refer to Section 1.3, Section 1.5, Section 2.6, Section 2.7.1, Section 2.7.2, Section 2.7.3 and Chapter 5)
30. Are related theory and hypotheses clearly reported?	Yes (Refer to Section 2.6, Section 2.7.1, Section 2.7.2, Section 2.7.3 and Chapter 5)
31. Are the data collection procedures presented, with relevant	Yes (Refer to Section 1.3, Section 3.4.3, Section 3.4.4,

Checklist Item	Addressed in this study?
motivation?	Section 3.4.5 and Chapter 4)
32. Is sufficient raw data presented (e.g. real life examples, quotations)?	No quotations included, in conformance with clauses of IBM's Confidentiality Agreement (Refer to Appendix X (Confidentiality Agreement))
33. Are the analysis procedures clearly reported?	Yes (Refer to Section 3.4.3, Section 3.4.4, Section 3.4.5 and Chapter 4)
34. Are threats to validity analyses reported along with countermeasures taken to reduce threats?	Yes (Refer to Chapter 4 and Section 6.3)
35. Are ethical issues reported openly (personal intentions, integrity issues, confidentiality)	Yes (Refer to Section 3.4.5.8 and Appendix X (Confidentiality Agreement))
36. Does the report contain conclusions, implications for practice and future research?	Yes (Refer to Section 6.1, Section 6.2 and Section 6.4)
37. Does the report give a realistic and credible impression?	Yes (Refer to thesis from Chapter 1 through Chapter 6)
38. Is the report suitable for its audience, easy to read and well structured?	Yes, findings from this case study were presented in multiple formats: (a) In multiple conferences proceedings (refer to Section 1.6), (b) In a submission to the I&ST Journal – this paper is under review (refer to Section 1.6), and (c) In the form of this thesis.

Construct Validity: Construct validity is associated with the adequacy with which variables represent the intended research construct of interest (Shadish, Cook, & Campbell, 2002). In this study communication was assessed based on messages sent around software tasks. These messages were extracted from Jazz and so may not represent all of the project teams' communications, which may have been facilitated through email, chat, and face-to-face communication for collocated team members. Offsetting this concern is the fact that, as Jazz was developed as a globally distributed project, developers were required to use messages so that all other contributors (irrespective of their physical location) were aware of product and process decisions regarding each WI (Cataldo & Ehrlich, 2012; Nguyen, Wolf, et al., 2008). Additionally, although messages sent around software tasks may be for many reasons outside of work (Aral & Walker, 2011), previous evidence confirms that communication during software engineering projects is generally related to work tasks (Shihab et al., 2010). Contextual analysis in this work also supports these authors' assessment (refer to Section 4.2 and Section 4.3).

Finally, in this work task change frequency was used to measure core developers' task performance (Cataldo & Herbsleb, 2008). However, all software tasks are not equal (Hackman, 1986); some tasks may be more computational and complex than others (e.g., a user experience task may not demand the same cognitive and mental rigor as that of a computational or coding intensive feature). While such complexities would likely 'even out' over the entire project, there is need for further research to take task complexity into account. That said, however, deeper analysis in this work provided

triangulation for the task change measure that was used as a proxy for task performance (refer to Chapter 4).

Internal Validity: Internal validity considers the control of potential interference from other variables beyond the main variables under consideration (Runeson & Host, 2009). Given that case studies are generally uncontrolled, such studies generally invoke internal validity concerns (Yin, 2003). While this work used archival data, and so threats related to socially desirable responding by participants (Layman et al., 2006) is eliminated, the archival data studied in this work were not originally prepared for the purpose of research. This poses a threat to this study's internal reliability (Runeson & Host, 2009). Similarly, staff turnover at IBM Rational could also impact the patterns that were noticed in this work. That said, however, while this work could not account for artefacts for all of the 146 practitioners for the full duration of the Jazz project during the confirmatory aspect of the analysis (refer to Section 4.1), artefacts contributed by the 15 core developers that were explored during the deeper linguistic and directed content analysis stages of this work reflect software activities undertaken over the entire project duration (refer to Section 4.2 and Section 4.3). Multiple studies that have researched artefacts in the Jazz repository have also confirmed the representativeness and appropriateness of these artefacts for studying the software development process (Cataldo & Ehrlich, 2012; Ehrlich & Cataldo, 2012; Nguyen, Schroter, et al., 2008; Schroter, 2010).

External Validity: External validity considers the extent to which the findings of research may generalise to the domain under consideration (Yin, 2003). This issue has always been a source of contention for those criticising case studies, and so, is often countered through the application of rigorous and systematic study techniques (Runeson & Host, 2009) (refer to Table 22 for evaluation taxonomy). Although data saturation was achieved after analysing the third project case during the SNA phase of the analysis (refer to Section 3.4.2 and Appendix II), the tasks, history logs and messages from the ten teams (out of 94) may not necessarily represent all the teams' processes in the repository. Additionally, the work processes at IBM Rational are specific to this organisation and may not represent the organisation dynamics in other software development establishments. The software teams studied in this work used Jazz for project execution, including project management, project communication and software building (coding), and followed specific software methods. These software processes may be largely mature when compared to other small software organisations or open

source project environments. Thus, given that other smaller software teams may not create software in a similar environment to IBM's, the results shared in this work may not necessarily generalise to those software development situations.

Furthermore, as noted in Section 1.4, cultural differences and distance (geographical and temporal) may directly affect software development teams' performance (Espinosa et al., 2006), and these variables may also have an impact on team members' behaviors - which in turn may lead to performance issues (Jaanu et al., 2012). However, research examining the effects of cultural differences in global software teams has found few cultural gaps and behavioral differences among software practitioners from, and operating in, Western cultures, with the largest negative effects observed between Asian and Western cultures (Espinosa et al., 2006). Given that the teams (and core developers) studied in this work all operated in Western cultures, this issue may have had little effect on the patterns of behaviors that were observed.

That said, Costa et al. (2011) confirmed that practitioners of the Jazz project exhibited similar coordination needs to practitioners of four projects operating in two distinct companies. Further, Potts (1993) noted that issues present in large industrial projects are likely to be representative of phenomena elsewhere in comparable contexts. However, it is not possible for case studies to provide completely generalisable findings (Potts, 1993; Runeson & Host, 2009). Thus, this work provides conjectures, and encourages future research (outlined in Chapter 5 and Section 6.4.2). There is also optimism, and particularly given the consistency in coordination requirements that was noted for other similar projects (Costa et al., 2011), that the results in this work may be applicable to similar large-scale distributed software development endeavours.

Reliability: Reliability assesses the repeatability of research, and the likelihood of other researchers being able to replicate the study findings using the identical study design and procedures (Runeson & Host, 2009). Benchmarks are generally used to limit threats to reliability (Layman et al., 2006). In this study context, SNA techniques are established approaches used for studying communication among individuals across many disciplines (De Laat et al., 2007; Wasserman & Faust, 1997; Willging, 2005), and this approach was also used previously for studying software teams' interactions as was undertaken in this work (Cataldo et al., 2006; Wolf, Schroter, Damian, Panjer, et al., 2009). Additionally, the LIWC language constructs that were used in this work to measure attitudes have been utilized previously to investigate this subject and were

assessed for validity and reliability (Li & Chignell, 2010; Mairesse et al., 2007; Pennebaker et al., 2007; Pennebaker & King, 1999; Yee et al., 2010).

However, although the LIWC dictionary was able to capture 66% of the overall words used by Jazz practitioners, the adequacy of these constructs in the specific context of software development warrants further investigation (Rigby & Hassan, 2007). To that end, a small sample of the messages were checked to see what might account for the remaining words being ignored by the LIWC tool and it was discovered that there were large amounts of cross referencing to other WIs in the messages, along with large amounts of highly specialized software related language (e.g., J2EE, LDAP, JACC, API, XML, TAME, JASS, Jazz, URI, REST, HTTP, Servlet, WIKI, UseCase, HTML, CVS, Dump, Config, SourceControl) evident in Jazz practitioners' exchanges (Licorish & MacDonell, 2013d). Their non-consideration here is therefore not a problem. Moreover, what was of interest, and was captured by the LIWC tool, was evidence of attitude, demeanour and behaviour. Additionally, triangulation generally provides a form of countermeasure against reliability and validity threats (Runeson & Host, 2009). Given that the linguistic analysis results in this work were also triangulated with positive and significant correlation results from the directed content analysis (refer to Section 4.3), the non-consideration of the terms noted above does not present a threat to the reliability of this study's findings.

Finally, directed CA, as conducted in this work and involving interpretation of textual data, is subjective, and so questions may naturally arise regarding the validity and reliability of the outcomes that are derived through its use. In this work multiple strategies were employed to mitigate these issues. First, the protocol that was adopted to study Jazz practitioners' interactions was created from those previously employed and tested in the study of interaction and knowledge sharing (Henri & Kaye, 1992; Zhu, 1996), and so there is a strong theoretical basis for its use. Second, this protocol was piloted and extended by deriving additional codes directly from the Jazz data, and the extended protocol was tested for accuracy, precision and objectivity, receiving an inter-rater measure indicative of excellent agreement (Hsieh & Shannon, 2005). Runeson & Host (2009) noted that such measures sufficiently validate case study reliability.

Overall then, while there are indeed potential threats to the findings derived from the research conducted and reported here, extensive effort has been expended to ensure that

the findings are as robust as possible. The implications of this case study's findings are considered next (in Section 6.4).

6.4 Research Implications

Findings in this work have implications for software engineering practice and future research. These details are provided in this section. Section 6.4.1 first presents the implications of this study's findings for software engineering practice, comprising both recommendations for project governance and collaboration and process tool enhancements. Section 6.4.2 then outlines the implications for software engineering research, including recommendations for future research and suggestions for those studying software artefacts and change logs.

6.4.1 Implications for SE Practice

This section presents the implications of this work's findings for software engineering practice, as outlined in two parts. The first part (Section 6.4.1.1) outlines implications for software project governance. Section 6.4.1.2 then provides the implications of this study's findings for collaboration and process tools.

6.4.1.1 Software Project Governance

Evidence in this work shows that Jazz globally distributed agile software development teams communicated most at project start-up and towards project completion. Given this pattern, Jazz project managers, or those governing projects where similar communication patterns are noted, should plan for these period of intensive activity by making adequate communication channels available. This requirement may be particularly necessary for software developments efforts conducted in distributed contexts, where time zone differences have been found to induce delays.

All of the ten Jazz teams studied in this work exhibited centralised communication patterns, and a small number of individuals dominated project communication networks, suggesting that project managers in globally distributed agile software development settings should not be alarmed if there is unevenness among team members' communications during software development. In fact, more dense communication networks may indicate a need for increased project coordination. Thus, project managers should be vigilant and prepared to closely manage their teams'

workload should this evidence appear. It was also observed that communication networks with low density did not affect team members' accessibility.

Findings in this work show that initial communication patterns were maintained throughout the software development project for Jazz practitioners. While this pattern may not be universal, software project managers should be vigilant of their project(s) communication demands and foster appropriate environments very early in the project. Changes in communication patterns may also be taken as a sign to closely examine project collaboration environments. For instance, for the more active practitioners, reduced communication may be a sign of reduced motivation or interest; on the other hand, excessive communication by the less active members may indicate that such individuals are overburdened or involved with unusually high amounts of features.

Additionally, in making provision for core developers' (or any other practitioners') absence or sudden withdrawal from the team, project management in globally distributed agile software development teams may promote team configurations that are likely to provide failsafe mechanisms – highly connected networks through task assignment. These networks are likely to reduce the threat imposed by the loss of key team players, and with them, the team's tacit knowledge.

Findings in this work revealed that formal team role assignment did not limit Jazz core developers' contributions. Thus, if the project environment is favourable globally distributed software development teams may self-organise successfully and work across roles to deliver successful outcomes – and this is likely to have a positive impact on team performance. Project managers should encourage their members to adapt and enact their natural roles; such that, programmers should not be seen entirely as solution providers and coders - persons occupying this role also need to communicate (and may lead coordination efforts) in order to succeed.

In fact, project managers employing rigid approaches may face challenges when managing teams such as those studied in this work, and strict project oversight tactics may be detrimental in globally distributed agile development environments. Evidence in this work suggests that the formal project leader in these settings should be willing to compromise, and perhaps encourage top performers, by releasing project control to these informal leaders as the need arises. That said, project managers should also be vigilant and prepared for instances where core developers may be unwilling to accept

informal project leadership and champion responsibilities. Specific incentives and rewards may be offered to those members to encourage such forms of participation, presuming of course that these members demonstrate this inclination early on.

These informal leaders, and central communicators in general, should cultivate less individualistic and more achievement-driven team norms. This would propagate to the entire team and may help with team performance. Evidence of negative and cynical attitudes during globally distributed agile software development should be discouraged, and particularly if these attitudes are observed among core developers. Given that these members occupy the heart of their teams' development activities, their expression of negative emotion may quickly circulate to the entire team and lead to disharmony and a tense team atmosphere. Such a project environment may affect overall team performance. On the other hand, a project environment that promotes social and task-based attitudes and behaviours seems ideal for team performance. This evidence was noticed in this work for all of the ten Jazz teams that were studied. Thus, globally distributed team leaders are encouraged to promote this form of team atmosphere. In fact, evidence in this work also revealed overall that Jazz developers were highly task focused. These findings suggest that while a positive and social group atmosphere may be ideal for maintaining a pleasant team ambience, a task-driven team focus (and roles) is important for globally distributed agile software developments, and so this should also be encouraged.

In terms of top software performers' changes in attitudes and knowledge sharing behaviours, Jazz core developers exhibited high levels of insight at project start-up, which may benefit weaker team members, and so it would be prudent for their project managers to implement strategies aimed at encouraging the engagement of less active developers at project inception. Although this pattern may be specific to IBM Rational, project leaders in other software organisations may also encourage less active members' participation during periods when their core developers are most active. This may benefit weaker members both in terms of knowledge acquisition and mentoring.

Jazz core developers were integral to project awareness, task coordination, idea generation and future planning, and these members were unhappy when they were least involved with task changes and team communication. This may be due to general dissatisfaction with the feedback that is provided by the wider team during their period of less prominence. Thus, keen participation (including clear communication) and

availability of the wider team may go some way to reducing tension and coordination breakdowns and enhancing team spirit around project completion for core developers. Strategies aimed at surrounding core developers with competent communicators would also help core developers to become quickly familiar with task knowledge during their periods of reduced presence. Further, access to multiple communication channels may also be important in reducing such dissatisfaction during globally distributed agile software developments.

Evidence in this work shows that technical needs drove Jazz core developers' knowledge sharing behaviours; such that, task involvement appears to have a general influence on core developers' need to provide knowledge and information to their teammates. Thus, communication hubs may not necessarily be formally denoted as communication and coordination specialists. These individuals communicate because of their actual development portfolio. Project managers should be vigilant that those that are integrally involved with task performance also need to communicate, and particularly in environments where teams work on individual software tasks, before then integrating these to a central software product. Lack of communication by these members may result in tacit knowledge loss should these members leave, but more critically, this may also result in lack of overall general team awareness and consequent integration issues.

Additionally, while those individuals who were highly active in their teams' communication network and task performance appeared critical to the functioning of their teams, project managers should be vigilant about the team's possible over-reliance on these members, which may negatively affect the quality of knowledge core developers are able to provide. In fact, these members' language process may provide project governance indicators. This work discovered correlation between Jazz core developers' use of work-related language and their task performance. While these results may be specific to Jazz teams, profound use of this form of speech by core developers may be a signal for the less active practitioners to reduce their level of reliance on these members. Additionally, evidence of this form of language process may also result in the provision of support for core developers, should they need such assistance. Furthermore, excessive use of work-related utterances may also stimulate project managers' interest, and serve as a warning sign that project leaders should be vigilant about the quality of the feedback that core developers are likely to provide the general team at this time.

Similar project governance pointers may be provided for core developers' contribution of knowledge sharing behaviours. For instance, this work revealed that when there were higher levels of questioning from Jazz core developers these practitioners were less active in task (WI) changes, tending to rely more on their wider teams. Thus, strategies aimed at surrounding core developers with competent communicators would help these practitioners to quickly become familiar with task knowledge. This would in turn promote overall shared team understanding, and reduced incidence of coordination breakdowns. This may be particularly useful for team synergies, as while it is not entirely clear from the results that were revealed in this work why core developers demonstrated reduced presence at specific times of their project, it was evident that these individuals expressed more unhappiness when they communicated many questions. The opposite was noted for other forms of knowledge sharing behaviours.

The aforementioned discussions comprise implications of this work's findings for software project governance. Discussions above include specific recommendations for improving the governance and potential performance of globally distributed agile software teams. Many of these recommendations may also apply more generally to software development as a whole. The next section (Section 6.4.1.2) extends these discussions, and considers the implication of the findings in this study in relation to collaboration and process tools.

6.4.1.2 Collaboration and Process Tools

Contextual analysis in this work shows that one half of Jazz practitioners' communications were directed at information sharing. This form of utterance, although useful for providing context awareness (i.e., explanations and information about software features, e.g., details about the outcomes of software builds), may not be as critical to the teams' development portfolios as the provision of questions, answers, suggestions and ideas. These latter, more critical types of communication may become 'lost' underneath the less significant messages (e.g., those expressing gratitude or praise). This issue was previously experienced by those involved in global software development, resulting in negative performance issues (Damian, Izquierdo, et al., 2007). Thus, including a message tagging feature in Jazz or any similar tool (as is done for tagging software tasks) could help developers to manage this wealth of communication. During time-constrained work periods, comment tags should help practitioners to identify and consider the most critical issues first. For instance, if comment tags were

labelled to express similar meanings to the categories and related scales in Table 6, a programmer coming in to work would likely review and action messages with Scales 9 (Instructions) and 1 (Questions) first, before going through the other messages in his or her order of preference.

During this work it was discovered that core developers' communication increased with higher levels of task involvement and their attitudes and knowledge sharing behaviours varied over the phases of their project. The link between communication and task involvement suggests that tools may provide useful visualizations of measures of development and coordination carried out by practitioners during the software development process. These project metrics may be compared against perceived or projected coordination measures in support of team management (Borici, Blincoe, Schroter, Valetto, & Damian, 2012). A project manager could use such tools in a similar way that project management and tracking tools are used for monitoring project performance. Of course, these tools would need to be sufficiently informed about the linkage between practitioners' involvement in task changes and communication.

Process tools that capture practitioners' communications (such as IBM Rational Jazz) may also benefit from attitude and behaviour visualizations. In order to be reliable and accurate, such tools will need to adhere to data mining principles (particularly data pre-processing techniques) and tested natural language processing methods (Licorish & MacDonell, 2013d). These tools would help project managers with team composition and task assignment (Licorish et al., 2009b). For instance, in monitoring teams' attitudes over the course of their project a relatively high incidence of negative words (e.g., afraid, hate, suck, dislike) expressed among developers would be an indicator of frustration and dissatisfaction. Such an observation may kick start human resource interventions (e.g., deeper interviews which lead to increasing the complement of highly skilled developers, staff rotation or some form of team-building activity). Similarly, in accommodating the findings for the relationship between core developers' expression of work-related terms and their involvement in task changes, evidence of such work focus attitudes during behaviour visualisations could potentially inform project governance interventions such as reducing less active practitioners' reliance on these members or the provision of assistance in the form of additional keen developers.

That said however, this work did not find irrefutable evidence of this latter linkage. Thus, this issue could benefit from additional research before its firm usage in software project diagnostics. This issue is considered further in Section 6.4.2.

6.4.2 Implications for SE Research (Future Work)

Software repositories hold details of teams' interactions that provide understandings of the way team processes are enacted during the software development life cycle. In particular, analysis opportunities presented by software artefacts are especially novel because of the reduction of the likely bias that arises from self-reporting and the unobtrusive nature of the investigation of behaviour from such sources (Licorish & MacDonell, 2012). Such understandings have led to numerous recommendations for improving software project performance (Abreu & Premraj, 2009; Bird et al., 2006a; Licorish & MacDonell, 2013b; Shihab et al., 2009). This work used repository sources to study the collaboration patterns of successful globally distributed agile teams, and how and why core developers contribute to globally distributed agile team dynamics.

SNA conducted in this research revealed that teams with centralised and low density communication networks may also remain highly connected through task assignment. From this evidence it was suggested that such highly connected communication networks may reduce risks related to knowledge loss should members leave such teams. However, there still remain questions around the effectiveness of such a strategy for dealing with knowledge transfer. Accordingly, future research is encouraged to study this issue.

This research has also revealed that Jazz's core developers worked across multiple roles, and these members were crucial to their teams' organizational, intrapersonal and interpersonal processes. Additionally, although these individuals were highly task- and achievement-focused, they also contributed to the maintenance of a positive team atmosphere. In fact, these members were eager for their teams to succeed, to the extent that they expressed unhappiness when they were least involved in task (feature) changes, at a time when they also asked more questions. Given that most of these members occupied the formal programmer role, there is little doubt that these members were intrinsically driven (Chang et al., 2013; Kline & Peters, 1991). While the literature has shed some light on this issue (Allen & Meyer, 1990; Chang et al., 2013; Kline & Peters, 1991; Lee & Xia, 2005; Morgan & Hunt, 1994), there is need for future research

to investigate core developers' motivation, perhaps through face-to-face research mechanisms (e.g., interviews).

Results obtained in this work show that core developers made reduced contributions to communication and task performance towards project completion. While this evidence was previously linked to the increasing level of task difficulty encountered as software projects progress, the trend of the results revealed in this work over the first three project phases did not support this assessment (refer to Section 6.1.3 for retrospections). It may instead be contended that core developers were involved in other work outside of the repository towards project completion. Future research employing complementary interview-related techniques would again help to provide additional insights into the reasons for this pattern.

Finally, results in this work revealed that core developers were most active in task performance when they used more work-related terms. These results provide some support for the linkage between attitudes and team performance, as established in social motivation theories (Geen, 1991; Inkpen & Tsang, 2005; Levin & Cross, 2004), and particularly for the way certain behavioural norms are said to encourage individual performance (Denning, 2012; Quigley et al., 2007). However, overall, the results in this work were not definitive; this question would benefit from additional research, comprising a much larger sample of artefacts.

Accordingly, the following open questions are outlined to address these issues: Do communication networks connected through task assignment provide an effective strategy for dealing with knowledge transfer/loss? What ignites core developers' motivation? Why are core developers most negative and self-focused at project completion? What is responsible for core developers' reduced task performance and communication towards project completion? Research may also consider the research questions in this study (refer to Chapter 2) in relation to the quality of core developers' deliverables (e.g., How do core developers' attitudes and knowledge sharing behaviours affect the quality of the features they deliver?). Further research is also needed for testing the model that is provided by this work (refer to Figure 31) with other AGSD teams.

From a study design perspective, the value of employing more contextual analysis techniques to understand team processes cannot be overstated. Quantitative and

qualitative techniques may indeed complement each other to provide richer accounts of team phenomena. The value of the balance provided by these approaches was exhibited during this work. This study overcame the limitations of purely quantitative approaches that ignore the complexities of human psychology, and the time-intensive and potentially invasive nature of field work required in full case studies. Furthermore, previous work has placed significant emphasis on the need for longitudinal studies to understand changes in teams' activities over time, as against the frequently used snapshot or cross-sectional analysis approaches. The utilization of deeper analysis techniques in this longitudinal study shows that studying software practitioners' behaviours, even at the word usage level, provides enhanced understanding of SE teams. Thus, researchers are encouraged to conduct such temporal analyses and to triangulate frequency-based approaches with contextual analysis techniques when examining teams' behavioural issues.

References

- Abrahamsson, P., Marchesi, M., Succi, G., Sfetsos, P., Stamelos, I., Angelis, L. (2006). Investigating the Impact of Personality Types on Communication and Collaboration-Viability in Pair Programming – An Empirical Study. In *Extreme Programming and Agile Processes in Software Engineering* (Vol. 4044, pp. 43-52): Springer Berlin / Heidelberg. doi:10.1007/11774129_5.
- Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003, 3-10 May 2003). *New directions on agile methods: a comparative analysis*. presented at the meeting of the Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon. doi:10.1109/ICSE.2003.1201204.
- Abreu, R., & Premraj, R. (2009, August 24-28, 2009). *How developer communication frequency relates to bug introducing changes*. presented at the meeting of the Joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops IWPSE/Evol 09 Amsterdam, The Netherlands. doi:10.1145/1595808.1595835.
- Acuna, S. T., Gomez, M., & Juristo, N. (2009). How do personality, team processes and task characteristics relate to job satisfaction and software quality? *Information and Software Technology*, 51(3), 627-639. doi:10.1016/j.infsof.2008.08.006.
- Acuna, S. T., & Juristo, N. (2004). Assigning people to roles in software projects. *Software Practice and Experience*, 34(7), 675-696. doi:10.1002/spe.586.
- Acuna, S. T., Juristo, N., & Moreno, A. M. (2006). Emphasizing human capabilities in software development. *IEEE Software*, 23(2), 94-101.
- Adams, S. L., & Anantatmula, V. (2010). Social and behavioral influences on team process. *Project Management Journal*, 41(4), 89-98. doi:10.1002/pmj.20192.
- Ahuja, M. K., Galletta, D. F., & Carley, K. M. (2003). Individual Centrality and Performance in Virtual R&D Groups: An Empirical Study. *Management Science*, 49(1), 21-38. doi:10.1287/mnsc.49.1.21.12756.
- Al-Ani, B., Horspool, A., & Bligh, M. C. (2011). Collaborating with ‘virtual strangers’: Towards developing a framework for leadership in distributed teams. *Leadership*, 7(3), 219-249. doi:10.1177/1742715011407382.
- Al-Rawas, A., & Easterbrook, S. (1996, February 1-2, 1996). *Communication problems in requirements engineering: a field study*. presented at the meeting of the Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering, Royal Society, London.
- Allen, N. J., & Meyer, J. P. (1990). The measurement and antecedents of affective, continuance and normative commitment to the organization. *Journal of Occupational Psychology*, 63(1), 1-18. doi:10.1111/j.2044-8325.1990.tb00506.x.
- Andre, M., Baldoquin, M. G., & Acuna, S. T. (2011). Formal model for assigning human resources to teams in software projects. *Information and Software Technology*, 53(3), 259-275. doi:10.1016/j.infsof.2010.11.011.
- Antoniol, G., Ayari, K., Penta, M. D., Khomh, F., & Gueheneuc, Y.-G. (2008, October 27-30, 2008). *Is it a bug or an enhancement?: a text-based approach to classify change requests*. presented at the meeting of the Proceedings of the 2008

- conference of the center for advanced studies on collaborative research: meeting of minds, Ontario, Canada. doi:10.1145/1463788.1463819.
- Aral, S., & Walker, D. (2011). Identifying Social Influence in Networks Using Randomized Experiments. *IEEE Intelligent Systems*, 26(5), 91-96. doi:10.1109/MIS.2011.89.
- Aranda, J., & Venolia, G. (2009, May 16-24, 2009). *The secret life of bugs: Going past the errors and omissions in software repositories*. presented at the meeting of the Proceedings of the 31st International Conference on Software Engineering, Vancouver, BC, Canada. doi:10.1109/icse.2009.5070530.
- Ardichvili, A., Page, V., & Wentling, T. (2003). Motivation and barriers to participation in virtual knowledge-sharing communities of practice. *Knowledge Management*, 7(1), 64-77.
- Arora, A., & Gambardella, A. (2005). *The Globalization of the Software Industry: Perspectives and Opportunities for Developed and Developing Countries*. Washington, DC: National Bureau of Economic Research.
- Ashforth, B. E. (2001). *Role transitions in organizational life: An identity based perspective*. Mahwah, NJ: Lawrence Earlbaum.
- Augustine, S., Payne, B., Sencindiver, F., & Woodcock, S. (2005). Agile project management: steering from the edges. *Communications of the ACM*, 48(12), 85 - 89. doi:10.1145/1101779.1101781.
- Aune, E., Bachmann, A., Bernstein, A., Bird, C., & Devanbu, P. (2008, November 9-14, 2008). *Looking back on prediction: A retrospective evaluation of bug-prediction techniques*. presented at the meeting of the Student Research Forum at SIGSOFT 2008/FSE 16, November 2008, Atlanta, GA.
- Aviv, R., Erlich, Z., Ravid, G., & Geva, A. (2003). Network analysis of knowledge construction in asynchronous learning networks. *Journal of Asynchronous Learning Networks*, 7(3), 1-23.
- Babar, M. A., Kitchenham, B., & Gorton, I. (2006, May 20-28, 2006). *Towards a distributed software architecture evaluation process: a preliminary assessment*. presented at the meeting of the Proceedings of the 28th international conference on Software engineering, Shanghai, China. doi:10.1145/1134285.1134430.
- Babbie, E. (2004). *The Practice of Social Research* (10th ed.). Belmont, CA: Wadsworth, Thomson Learning Inc.
- Bacchelli, A., Lanza, M., & Robbes, R. (2010, May 2-8, 2010). *Linking e-mails and source code artifacts*. presented at the meeting of the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, Cape Town, South Africa. doi:10.1145/1806799.1806855.
- Bachmann, A., & Bernstein, A. (2009, August 24-28, 2009). *Software process data quality and characteristics: a historical view on open and closed source projects*. presented at the meeting of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, Amsterdam, Netherlands. doi:10.1145/1595808.1595830.
- Baddoo, N., Hall, T., & Jagielska, D. (2006). Software developer motivation in a high maturity company: a case study. *Software Process: Improvement and Practice*, 11(3), 219-228. doi:10.1002/spip.265.

- Bales, R. F. (1950a). *Interaction Process Analysis: A Method for the Study of Small Groups*. Cambridge, MA: Addison Wesley.
- Bales, R. F. (1950b). A Set of Categories for the Analysis of Small Group Interaction. *American Sociological Review*, 15(2), 257-263.
- Ball, G., & Breese, J. (2000). Relating Personality and Behavior: Posture and Gestures. In *Affective Interactions* (Vol. 1814, pp. 196-203): Springer Berlin / Heidelberg. doi:10.1007/10720296_14.
- Banerjee, M., Capozzoli, M., McSweeney, L., & Sinha, D. (1999). Beyond kappa: A review of interrater agreement measures. *Canadian Journal of Statistics*, 27(1), 3-23. doi:10.2307/3315487.
- Barcellini, F., Detienne, F., Burkhardt, J.-M., & Sack, W. (2008). A socio-cognitive analysis of online design discussions in an Open Source Software community. *Interacting with Computers*, 20(1), 141-165. doi:10.1016/j.intcom.2007.10.004.
- Bavelas, A. (1950). Communication Patterns in Task-Oriented Groups. *The Journal of the Acoustical Society of America*, 22(6), 725 -730.
- Baysal, O., & Malton, A. J. (2007, May 20-26, 2007). *Correlating Social Interactions to Release History during Software Evolution*. presented at the meeting of the Fourth International Workshop on Mining Software Repositories, 2007. ICSE Workshops MSR '07, Minneapolis, MN. doi:10.1109/msr.2007.4.
- Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Reading, MA: Addison-Wesley Longman, Inc.
- Belbin, R. M. (2002). *Management teams: why they succeed or fail*. Woburn, UK: Butterworth-Heinemann.
- Benestad, H. C., & Hannay, J. E. (2012, September 17-22, 2012). *Does the prioritization technique affect stakeholders' selection of essential software product features?* presented at the meeting of the ACM-IEEE international symposium on Empirical software engineering and measurement, Lund, Sweden. doi:10.1145/2372251.2372300.
- Benne, K. D., & Sheats, P. (1948). Functional Roles of Group Members. *Journal of Social Issues*, 4(2), 41-49. doi:10.1111/j.1540-4560.1948.tb01783.x.
- Beranek, G., Zuser, W., & Grechenig, T. (2005, May 15-16, 2005). *Functional group roles in Software Engineering teams*. presented at the meeting of the Workshop on Human and Social Factors of Software Engineering, HSSE'05, St. Louis, Missouri, USA. doi:10.1145/1083106.1083108.
- Bereiter, C., & Scardamalia, M. (1987). *The Psychology of Written Composition*. Broadway, Hillsdale, NJ: Lawrence Erlbaum Associates.
- Berelson, B. (1952). *Content analysis in communication research* (Vol. #1 of Foundations of communications research). Glencoe, Illinois: Free Press.
- Berenbach, B., & Borotto, G. (2006, May 20-28, 2006). *Metrics for model driven requirements development*. presented at the meeting of the Proceedings of the 28th international conference on Software engineering, Shanghai, China. doi:10.1145/1134285.1134348.
- Bettenburg, N., Sascha, J., Schroter, A., Weib, C., Premraj, R., & Zimmermann, T. (2007, October 21-25, 2007). *Quality of bug reports in Eclipse*. presented at the meeting of the Proceedings of the 2007 OOPSLA workshop on eclipse

technology eXchange, Montreal, Quebec, Canada.
doi:10.1145/1328279.1328284.

- Bhat, T., & Nagappan, N. (2006, December 6-8, 2006). *Building Scalable Failure-proneness Models Using Complexity Metrics for Large Scale Software Systems*. presented at the meeting of the 13th Asia Pacific Software Engineering Conference, APSEC 2006, Bangalore, India. doi:10.1109/apsec.2006.25.
- Biddle, B. J., & Thomas, E. J. (1966). *Role theory: Concepts and research* (Vol. (Eds.)). New York: John Wiley & Sons.
- Bird, C., Gourley, A., Devanbu, P., Gertz, M., & Swaminathan, A. (2006a, May 22-23, 2006). *Mining email social networks*. presented at the meeting of the Proceedings of the 2006 international workshop on Mining Software Repositories, Shanghai, China. doi:10.1145/1137983.1138016.
- Bird, C., Gourley, A., Devanbu, P., Gertz, M., & Swaminathan, A. (2006b, May 22-23, 2006). *Mining email social networks in Postgres*. presented at the meeting of the Proceedings of the 2006 International workshop on Mining Software Repositories, Shanghai, China. doi:10.1145/1137983.1138033.
- Bird, C., Murphy, B., Nagappan, N., & Zimmermann, T. (2011, March 19-23, 2011). *Empirical software engineering at Microsoft Research*. presented at the meeting of the ACM 2011 Conference on Computer Supported Cooperative Work, Hangzhou, China. doi:10.1145/1958824.1958846.
- Bird, C., Nagappan, N., Devanbu, P., Gall, H., & Murphy, B. (2009, May 16-24, 2009). *Does distributed development affect software quality? An empirical case study of Windows Vista*. presented at the meeting of the Proceedings of the 31st International Conference on Software Engineering, Vancouver, BC. doi:10.1109/icse.2009.5070550.
- Bird, C., Nagappan, N., Gall, H., Murphy, B., & Devanbu, P. (2009, November 16-19, 2009). *Putting it all together: using socio-technical networks to predict failures*. presented at the meeting of the the 20th IEEE International Conference on Software Reliability Engineering, Bengaluru-Mysuru, India. doi:10.1109/ISSRE.2009.17.
- Blackman, M. C. (2002). The Employment Interview via the Telephone: Are We Sacrificing Accurate Personality Judgments for Cost Efficiency? *Journal of Research in Personality*, 36(3), 208-223. doi:10.1006/jrpe.2001.2347.
- Blanchette, J. (1999). Register choice: Linguistic variation in an on-line classroom. *International Journal of Educational Telecommunications*, 5(2), 127-142.
- Blaskovich, J. L. (2008). Exploring the Effect of Distance: An Experimental Investigation of Virtual Collaboration, Social Loafing, and Group Decisions. *Journal of Information Systems*, 22(1), 27-46.
- Bock, G. W., & Kim, Y.-G. (2002). Breaking the Myths of Rewards: An Exploratory Study of Attitudes about Knowledge Sharing. *Information Resources Management Journal*, 15(2), 14-21. doi:10.4018/irmj.2002040102.
- Boehm, B. W. (2006, May 20-28, 2006). *A view of 20th and 21st century software engineering*. presented at the meeting of the Proceeding of the 28th International Conference on Software Engineering, Shanghai, China. doi:10.1145/1134285.1134288.

- Boehm, B. W., & Basili, V. R. (2001). Software Defect Reduction Top 10 List. *Computer*, 34(1), 135-137. doi:10.1109/2.962984.
- Boehm, B. W., Clark, B., Horowitz, E., Westland, C., Madachy, R., & Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1(1), 57-94. doi:10.1007/bf02249046.
- Boehm, B. W., & Turner, R. (2003a). *Balancing Agility and Discipline: A guide for the Perplexed*. New York: Addison-Wesley.
- Boehm, B. W., & Turner, R. (2003b). Using risk to balance agile and plan-driven methods. *IEEE Journal*, 36(6), 57-66.
- Boh, W. F., Slaughter, S. A., & Espinosa, J. A. (2007). Learning from Experience in Software Development: A Multilevel Analysis. *Management Science*, 53(8), 1315-1331. doi:10.1287/mnsc.1060.0687.
- Borici, A., Blincoe, K., Schroter, A., Valetto, G., & Damian, D. (2012, June 2, 2012). *ProxiScientia: Toward real-time visualization of task and developer dependencies in collaborating software development teams*. presented at the meeting of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2012, Zurich, Switzerland. doi:10.1109/chase.2012.6223024.
- Boyle, J. M., Bury, K. F., & Evey, R. J. (1983). Two Studies Evaluating Learning and Use of QBE and SQL. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 27(7), 663-667. doi:10.1177/154193128302700732.
- Bretz, R. (1983). *Media for interactive communication*. London: Sage Publication.
- Brooks, A., Clarke, D., & McGale, P. A. (1994). Investigating stellar variability by normality tests. *Vistas in Astronomy*, 38(4), 377-399. doi:10.1016/0083-6656(94)90011-6.
- Brown, R. (2000). Social identity theory: past achievements, current problems and future challenges. *European Journal of Social Psychology*, 30(6), 745-778. doi:10.1002/1099-0992.
- Bryman, A. (1984). The Debate about Quantitative and Qualitative Research: A Question of Method or Epistemology? *The British Journal of Sociology*, 35(1), 75-92.
- Bullen, M. (1998). Participation and Critical Thinking in Online University Distance Education. *Journal of Distance Education*, 13(2).
- Calefato, F., Gendarmi, D., & Lanubile, F. (2009, August 24-28, 2009). *Embedding social networking information into jazz to foster group awareness within distributed teams*. presented at the meeting of the Proceedings of the 2nd international workshop on Social software engineering and applications, Amsterdam, The Netherlands. doi:10.1145/1595836.1595842.
- Capiluppi, A., Fernandez-Ramil, J., Higman, J., Sharp, H., & Smith, N. (2007, May 20-26, 2007). *An Empirical Study of the Evolution of an Agile-Developed Software System*. presented at the meeting of the 29th International Conference on Software Engineering, 2007 (ICSE 2007), Minneapolis, MN. doi:10.1109/icse.2007.14.
- Capretz, L. F., & Ahmed, F. (2010). Why do we need personality diversity in software engineering? *SIGSOFT Softw. Eng. Notes*, 35(2), 1-11. doi:10.1145/1734103.1734111.

- Carmel, E., & Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. *IEEE Software*, 18(2), 22-29. doi:10.1109/52.914734.
- Cataldo, M., Bass, M., Herbsleb, J. D., & Bass, L. (2007, August 27-30, 2007). *On Coordination Mechanisms in Global Software Development*. presented at the meeting of the Second IEEE International Conference on Global Software Engineering, 2007, Munich, Germany. doi:10.1109/icgse.2007.33.
- Cataldo, M., & Ehrlich, K. (2012, May 5-10, 2012). *The impact of communication structure on new product development outcomes*. presented at the meeting of the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Austin, Texas, USA. doi:10.1145/2207676.2208722.
- Cataldo, M., & Herbsleb, J. D. (2008, November 8-12, 2008). *Communication networks in geographically distributed software development*. presented at the meeting of the Proceedings of the 2008 ACM conference on Computer supported cooperative work, San Diego, CA, USA. doi:10.1145/1460563.1460654.
- Cataldo, M., Wagstrom, P. A., Herbsleb, J. D., & Carley, K. M. (2006, November 4-8, 2006). *Identification of coordination requirements: implications for the Design of collaboration and awareness tools*. presented at the meeting of the 20th anniversary conference on Computer Supported Cooperative Work, 06, Banff, Alberta, Canada. doi:10.1145/1180875.1180929.
- Chang, K., & Ehrlich, K. (2007, October 22-25, 2007). *Out of sight but not out of mind?: Informal networks, communication and media use in global software teams*. presented at the meeting of the Proceedings of the 2007 Conference of the center for advanced studies on Collaborative Research, Richmond Hill, Ontario, Canada. doi:10.1145/1321211.1321221.
- Chang, K., Yen, H., Chiang, C., & Parolia, N. (2013). Knowledge contribution in information system development teams: An empirical research from a social cognitive perspective. *International Journal of Project Management*, 31(2), 252-263. doi:10.1016/j.ijproman.2012.06.005.
- Charette, R. N. (2005). Why software fails. *Spectrum, IEEE*, 42(9), 42-49.
- Checkland, P. (2000). Soft systems methodology: a thirty year retrospective. *Systems Research and Behavioral Science*, 17(1), 11- 58. doi:10.1002/1099-1743.
- Cheng, L.-T., Hupfer, S., Ross, S., Patterson, J., Clark, B., & De Souza, C. (2003, October 26-30, 2003). *Jazz: a collaborative application development environment*. presented at the meeting of the Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, Anaheim, CA, USA. doi:10.1145/949344.949370.
- Chin, G. (2004). *Agile Project Management: How to Succeed in the Face of Changing Project Requirements*. New York: American Management Association.
- Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer*, 34(11), 131-133.
- Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Colomo-Palacios, R., Cabezas-Isla, F., Garcia-Crespo, A., & Soto-Acosta, P. (2010). Generic Competences for the IT Knowledge Workers: A Study from the Field. *Communications in Computer and Information Science*, 111, 1-7. doi:10.1007/978-3-642-16318-0_1.

- Coltheart, M. (1981). The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology*, 33(4), 497 - 505.
- Constant, D., Sproull, L., & Kiesler, S. (1996). The Kindness of Strangers: The Usefulness of Electronic Weak Ties for Technical Advice. *Organization Science*, 7(2), 119-135. doi:10.1287/orsc.7.2.119.
- Constas, M. A. (1992). Qualitative Analysis as a Public Event: The Documentation of Category Development Procedures. *American Educational Research Journal*, 29(2), 253-266. doi:10.3102/00028312029002253.
- Coram, M., & Bohner, S. (2005, April 4-7, 2005). *The impact of agile methods on software project management*. presented at the meeting of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS '05. , Greenbelt, Maryland. doi:10.1109/ECBS.2005.68.
- Costa, J. M., Cataldo, M., & de Souza, C. (2011, May 7-12, 2011). *The scale and evolution of coordination needs in large-scale distributed projects: implications for the future generation of collaborative tools*. presented at the meeting of the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vancouver, BC, Canada. doi:10.1145/1978942.1979409.
- Creswell, J. (1998). *Qualitative inquiry and research design: Choosing among five traditions*. Thousand Oaks, CA: Sage.
- Creswell, J. (2002). *Qualitative, Quantitative, and Mixed Methods Approaches* (2nd ed.). Thousand Oaks, CA: Sage Publication.
- Crowston, K., Annabi, H., Howison, J., & Masango, C. (2004, November 5, 2004). *Effective work practices for software engineering: free/libre open source software development*. presented at the meeting of the Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research, Newport Beach, CA, USA. doi:10.1145/1029997.1030003.
- Crowston, K., & Howison, J. (2006). Hierarchy and Centralization in Free and Open Source Software Team Communications. *Knowledge, Technology, and Policy*, 18(4), 65-85.
- Crowston, K., Li, Q., Wei, K., Eseryel, Y. U., & Howison, J. (2007). Self-organization of teams for free/libre open source software development. *Information and Software Technology*, 49(6), 564-575. doi:10.1016/j.infsof.2007.02.004.
- Crowston, K., Wei, K., Howison, J., & Wiggins, A. (2008). Free/Libre open-source software development: What we know and what we do not know. *ACM Computing Surveys*, 44(2), 1-35. doi:10.1145/2089125.2089127.
- Crowston, K., Wei, K., Li, Q., & Howison, J. (2006, January 4-7, 2006). *Core and Periphery in Free/Libre and Open Source Software Team Communications*. presented at the meeting of the Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 06, Kauai, HI, USA. doi:10.1109/hicss.2006.101.
- Cummings, J. N. (2004). Work Groups, Structural Diversity, and Knowledge Sharing in a Global Organization. *Management Science*, 50(3), 352-364. doi:10.1287/mnsc.1030.0134.

- Cummings, J. N., & Cross, R. (2003). Structural properties of work groups and their consequences for performance. *Social Networks*, 25(3), 197-210. doi:10.1016/S0378-8733(02)00049-7.
- Cunha, A. B., Canen, A. G., & Capretz, M. A. M. (2009, 17 May, 2009). *Personalities, cultures and software modeling: Questions, scenarios and research directions*. presented at the meeting of the Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, Vancouver, BC, Canada. doi:10.1109/chase.2009.5071406.
- Curtis, B. (1981). *Human factors in software development*. Piscataway, N.J.: IEEE Computer Society.
- Curtis, B., Hefley, W. E., & Miller, S. A. (2001). *People Capability Maturity Model Version 2.0 CMU/SEI-2001-MM-01*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.
- Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM*, 31(11), 1268-1287. doi:10.1145/50087.50089.
- Dabbish, L. A., Kraut, R., Fussell, S., & Kiesler, S. (2005, April 2-7, 2005). *Understanding email use: predicting action on a message*. presented at the meeting of the Proceedings of the SIGCHI conference on Human factors in computing systems, Portland, Oregon, USA. doi:10.1145/1054972.1055068.
- Daft, R. L., & Lengel, R. H. (1986). Organizational information requirements, media richness and structural design. *Management Science*, 32(5), 554-571. doi:10.1287/mnsc.32.5.554.
- Damerau, F., J. (1993). Generating and evaluating domain-oriented multi-word terms from texts. *Inf. Process. Manage.*, 29(4), 433-447. doi:10.1016/0306-4573(93)90039-g.
- Damian, D., Izquierdo, L., Singer, J., & Kwan, I. (2007, August 27-30, 2007). *Awareness in the Wild: Why Communication Breakdowns Occur*. presented at the meeting of the Proceedings of the International Conference on Global Software Engineering, Munich, Germany. doi:10.1109/icgse.2007.13.
- Damian, D., Kwan, I., & Marczak, S. (2010). Requirements-Driven Collaboration: Leveraging the Invisible Relationships between Requirements and People. In *Collaborative Software Engineering* (pp. 57-76): Springer Berlin Heidelberg. doi:10.1007/978-3-642-10294-3_3.
- Damian, D., Marczak, S., & Kwan, I. (2007, October 15-19, 2007). *Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks*. presented at the meeting of the 15th IEEE International Requirements Engineering Conference (RE '07), New Delhi, India. doi:10.1109/RE.2007.51.
- Damian, D., & Zowghi, D. (2003). Requirements engineering challenges in multi-site software development organisations. *Requirements Engineering*, 8(3), 149-160. doi:10.1007/s00766-003-0173-1.
- Danait, A. (2005, July 24-29, 2005). *Agile offshore techniques - a case study*. presented at the meeting of the Proceedings of the Agile Conference, 2005, Denver, USA. doi:10.1109/adc.2005.9.

- Datta, S., Kaulgud, V., Sharma, V. S., & Kumar, N. (2010, February 25-27, 2010). A *social network based study of software team dynamics*. presented at the meeting of the Proceedings of the 3rd India Software Engineering Conference, Mysore, India. doi:10.1145/1730874.1730883.
- Datta, S., Sindhgatta, R., & Sengupta, B. (2011, February 23-26, 2011). *Evolution of developer collaboration on the jazz platform: a study of a large scale agile project*. presented at the meeting of the Proceedings of the 4th India Software Engineering Conference, Thiruvananthapuram, Kerala, India. doi:10.1145/1953355.1953359.
- Datta, S., Sindhgatta, R., & Sengupta, B. (2012). Talk versus work: characteristics of developer collaboration on the jazz platform. *SIGPLAN Notices*, 47(10), 655-668. doi:10.1145/2398857.2384664.
- Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. *Management Science*, 35(8), 982-1003. doi:10.2307/2632151.
- Davis, M. S. (1971). That's Interesting!: Towards a Phenomenology of Sociology and a Sociology of Phenomenology. *Philosophy of the Social Sciences*, 1(2), 309-344. doi:10.1177/004839317100100211.
- De Dreu, C. K. W., & Weingart, L. R. (2003). Task Versus Relationship Conflict, Team Performance, and Team Member Satisfaction: A Meta-Analysis. *Journal of Applied Psychology*, 88(4), 741-749.
- De Laat, M., Lally, V., Lipponen, L., & Simons, R.-J. (2007). Investigating patterns of interaction in networked learning and computer-supported collaborative learning: A role for Social Network Analysis. *International Journal of Computer-Supported Collaborative Learning*, 2(1), 87-103. doi:10.1007/s11412-007-9006-4.
- De, P., Sinha, A. P., & Vessey, I. (2001). An empirical investigation of factors influencing object-oriented database querying. *Information Technology and Management*, 2(1), 71-93. doi:10.1023/a:1009934820999.
- de Souza, C. R. B., & Redmiles, D. F. (2009). On The Roles of APIs in the Coordination of Collaborative Software Development. *Computer Supported Cooperative Work (CSCW)*, 18(5-6), 445-475. doi:10.1007/s10606-009-9101-3.
- De Souza, S., Anquetil, N., & De Oliveira, K. (2005, 2005). A study of the documentation essential to software maintenance. *ACM Press, New York*. Symposium conducted at the meeting of the Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information, Coventry, United Kingdom.
- De Vries, R. E., Van den Hooff, B., & De Ridder, J. A. (2006). Explaining Knowledge Sharing: The Role of Team Communication Styles, Job Satisfaction, and Performance Beliefs. *Communication Research*, 33(2), 115-135. doi:10.1177/0093650205285366.
- De Wever, B., Schellens, T., Valcke, M., & Van Keer, H. (2006). Content analysis schemes to analyze transcripts of online asynchronous discussion groups: a review. *Computers and Education*, 46(1), 6-28. doi:10.1016/j.compedu.2005.04.005.
- Dekker, A. (2002). Applying Social Network Analysis Concepts to Military C4ISR Architectures. 24(3), 93-103.

- Dempsey, B., J. Weiss, D., Jones, P., & Greenberg, J. (2002). Who is an open source software developer? *Communications of the ACM*, 45(2), 67-72. doi:10.1145/503124.503125.
- Denning, P. J. (2012). Moods. *Communications of the ACM*, 55(12), 33-35. doi:10.1145/2380656.2380668.
- Denning, P. J., & Dunham, R. (2010). *The Innovator's Way*. Cambridge, MA: The MIT Press.
- Di Penta, M. (2012, June 3, 2012). *Mining developers' communication to assess software quality: Promises, challenges, perils*. presented at the meeting of the 3rd International Workshop on Emerging Trends in Software Metrics (WETSoM), 2012 Zurich, Switzerland. doi:10.1109/WETSoM.2012.6226987.
- Doty, D. H., & Glick, W. H. (1994). Typologies as a Unique Form of Theory Building: Toward Improved Understanding and Modeling. *The Academy of Management Review*, 19(2), 230-251. doi:10.2307/258704.
- Downey, J. (2009). Designing Job Descriptions for Software Development. In *Information Systems Development: Challenges in Practice, Theory, and Education*. (pp. 447-460). USA: Springer US. doi:10.1007/978-0-387-68772-8_34.
- Druskat, V. U., & Wolff, S. B. (2001). Building the emotional intelligence of groups. *Harvard Business Review*, 79(3), 80.
- Dubin, R. (1978). *Theory Building*. (Vol. Revised). London: Free Press.
- Ducheneaut, N. (2005). Socialization in an Open Source Software Community: A Socio-Technical Analysis. *Computer Supported Cooperative Work (CSCW)*, 14(4), 323-368. doi:10.1007/s10606-005-9000-1.
- Dullemond, K., Gamen, B. v., & Solingen, R. v. (2009, July 13-16, 2009). *How Technological Support Can Enable Advantages of Agile Software Development in a GSE Setting*. presented at the meeting of the Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering, Limerick, Ireland. doi:10.1109/icgse.2009.22.
- Dutoit, A. H., & Bruegge, B. (1998). Communication Metrics for Software Development. *IEEE Trans. Softw. Eng.*, 24(8), 615-628. doi:10.1109/32.707697.
- Earl, M. J. (1996). The Risks of Outsourcing IT. *MIT Sloan Management Review*, Spring 1996.
- Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting Empirical Methods for Software Engineering Research. In *Guide to Advanced Empirical Software Engineering* (pp. 285-311). London: Springer-Verlag. doi:10.1007/978-1-84800-044-5_11.
- Edwards, H., Puckett, R., & Jolly, A. (2006, June 26-29, 2006). *Analyzing communication patterns in software engineering projects*. presented at the meeting of the Software Engineering Research and Practice, SERP 2006, Las Vegas, Nevada, USA.
- Ehrlich, K., & Cataldo, M. (2012, February 11-15, 2012). *All-for-one and one-for-all?: a multi-level analysis of communication patterns and individual performance in geographically distributed software development*. presented at the meeting of the Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, Seattle, Washington, USA. doi:10.1145/2145204.2145345.

- Ehrlich, K., & Chang, K. (2006, October 16-19, 2006). *Leveraging expertise in global software teams: Going outside boundaries*. presented at the meeting of the Proceedings of the IEEE International Conference on Global Software Engineering, Florianopolis, Brazil. doi:10.1109/ICGSE.2006.261228.
- Ehrlich, K., Helander, M., Valetto, G., Davies, S., & Williams, C. (2008, May 10, 2008). *An Analysis of Congruence Gaps and Their Effect on Distributed Software Development*. presented at the meeting of the Socio-Technical Congruence Workshop at ICSE conference, Leipzig, Germany.
- Ehrlich, K., Lin, C.-Y., & Griffiths-Fisher, V. (2007, November 4-7, 2007). *Searching for experts in the enterprise: combining text and social network analysis*. presented at the meeting of the Proceedings of the 2007 international ACM Conference on Supporting Group Work, Sanibel Island, Florida, USA. doi:10.1145/1316624.1316642.
- El Emam, K., & Koru, A. G. (2008). A Replicated Survey of IT Software Project Failures. *IEEE Software*, 25(5), 84-90.
- Erlin, B. Y., Yusof, N., & Rahman, A. A. (2008, August 26-28, 2008). *Integrating Content Analysis and Social Network Analysis for analyzing Asynchronous Discussion Forum*. presented at the meeting of the International Symposium on Information Technology, 2008. ITSIM 2008, Kuala Lumpur. doi:10.1109/ITSIM.2008.4631996.
- Espinosa, J. A., DeLone, W., & Lee, G. (2006). Global boundaries, task processes and IS project success: a field study. *Information Technology & People*, 19(4), 345 - 370. doi:10.1108/09593840610718036.
- Espinosa, J. A., Slaughter, S. A., Kraut, R. E., & Herbsleb, J. D. (2007). Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development. *Organization Science*, 18(4), 613-630. doi:10.1287/orsc.1070.0297.
- Fahy, P. J., Crawford, G., & Ally, M. (2001). Patterns of interaction in a computer conference transcript. *The International Review of Research in Open and Distance Learning*, 2(1).
- Feldt, R., Angelis, L., Torkar, R., & Samuelsson, M. (2010). Links between the personalities, views and attitudes of software engineers. *Information and Software Technology*, 52(6), 611-624. doi:10.1016/j.infsof.2010.01.001.
- Fitzgerald, B., & Howcroft, D. (1998, December 13-16, 1998). *Competing dichotomies in IS research and possible strategies for resolution*. presented at the meeting of the Proceedings of the International Conference on Information Systems, Helsinki, Finland. doi:10.1145/353053.353066.
- Flyvbjerg, B. (2006). Five Misunderstandings About Case-Study Research. *Qualitative Inquiry*, 12(2), 219-245. doi:10.1177/1077800405284363.
- Franca, A. C. C., & Da Silva, F. Q. B. (2009, October 15-16, 2009). *An empirical study on software engineers motivational factors*. presented at the meeting of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Florida, USA. doi:10.1109/ESEM.2009.5316011.
- French, A., & Layzell, P. (1998, November 16-20, 1998). *A study of communication and cooperation in distributed software project teams*. presented at the meeting of the Proceedings of the International Conference on Software Maintenance, 1998, Bethesda, MD. doi:10.1109/icsm.1998.738503.

- Frost, R. (2007). Jazz and the Eclipse Way of Collaboration. *IEEE Software*, 24(6), 114-117. doi:10.1109/ms.2007.170.
- Funder, D. C., & Colvin, R. C. (1988). Friends and Strangers: Acquaintanceship, Agreement, and the Accuracy of Personality Judgment. *Journal of Personality & Social Psychology*, 55(1), 149-158.
- Gacek, C., & Arief, B. (2004). The many meanings of open source. *IEEE Software*, 21(1), 34-40. doi:10.1109/ms.2004.1259206.
- Gall, M. D., Borg, W. R., & Gall, J. P. (1996). *Educational Research: An Introduction*. White Plains, N.Y: Longman Publishers.
- Galliers, R. D., & Land, F. F. (1987). Viewpoint: choosing appropriate information systems research methodologies. *Communications of the ACM*, 30(11), 901-902. doi:10.1145/32206.315753.
- Gallivan, M. J. (2001). Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies. *Information Systems Journal*, 11(4), 277-304. doi:10.1046/j.1365-2575.2001.00108.x.
- Gaye, K., Butler, T., & Finnegan, P. (2010). Coordinating Global Virtual Teams: Building Theory from a Case Study of Software Development. In *Advanced Information Systems Engineering* (Vol. 6051, pp. 281-295). Heidelberg: Springer Berlin. doi:10.1007/978-3-642-13094-6_23.
- Geen, R. G. (1991). Social Motivation. *Annual Review of Psychology*, 42(1), 377-399. doi:10.1146/annurev.ps.42.020191.002113.
- Ghoniem, M., Fekete, J.-D., & Castagliola, P. (2005). On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2), 114-135. doi:10.1057/palgrave.ivs.9500092.
- Giddens, A. (1979). *Central Problems in Social Theory: Action, Structure and Contradiction in Social Analysis*. London: Macmillan.
- Giddens, A. (1984). *The constitution of society: outline of the theory of structuration*. Berkeley and Los Angeles: University of California Press.
- Giles, H., & Wiemann, J. M. (1993). Social Psychological Studies of Language: Current Trends and Prospects. *American Behavioral Scientist*, 36(3), 262-270. doi:10.1177/0002764293036003002.
- Gill, A. J., & Oberlander, J. (2002, August 7-10, 2002). *Taking care of the linguistic features of Extraversion*. presented at the meeting of the Paper presented at the 24th Annual Conference of the Cognitive Science Society, Fairfax, VA.
- Gill, A. J., & Oberlander, J. (2003, July 31-August 2, 2003). *Perception of email personality at zero-acquaintance: Extraversion takes care of itself; Neuroticism is a worry*. presented at the meeting of the Proceedings of the 25th Annual Conference of the Cognitive Science Society, Boston, Massachusetts.
- Gill, A. J., Oberlander, J., & Austin, E. (2006). Rating e-mail personality at zero acquaintance. *Personality and Individual Differences*, 40(3), 497-507.
- Glaser, B. G., & Strauss, A. L. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago: Aldine Publishing Company.

- Glass, R. L., Vessey, I., & Ramesh, V. (2002). Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44(8), 491-506. doi:10.1016/S0950-5849(02)00049-6.
- Godfrey-Smith, P. (2003). *Theory and Reality: an introduction to the philosophy of science*. Chicago: University of Chicago Press.
- Goguen, J. A. (1993, January 4-6, 1993). *Social issues in requirements engineering*. presented at the meeting of the Proceedings of IEEE International Symposium on Requirements Engineering, San Diego, CA. doi:10.1109/isre.1993.324858.
- Goldberg, L. R. (1981). Language and individual differences: The search for universals in personality lexicons. *Review of Personality and Social Psychology*, 2(1), 141-165.
- Gorla, N., & Lam, Y. W. (2004). Who should work with whom?: building effective software project teams. *Commun. ACM*, 47(6), 79-82. doi:10.1145/990680.990684.
- Gorunescu, F. (2011). Introduction to Data Mining. In J. Kacprzyk & L. C. Jain (Eds.), *Data Mining* (Vol. 12, pp. 1-43). Heidelberg: Springer Berlin. doi:10.1007/978-3-642-19721-5_1.
- Gosling, S. D., Rentfrow, P. J., & Swann, W. B. (2003). A very brief measure of the Big-Five personality domains. *Journal of Research in Personality*, 37(6), 504-528.
- Graesser, A. C., & Person, N. K. (1994). Question Asking During Tutoring. *American Educational Research Journal*, 31(1), 104-137. doi:10.3102/00028312031001104.
- Greenblatt, D., & Waxman, J. (1978, August 2-3, 1978). *A Study of Three Database Query Languages*. presented at the meeting of the Proceedings of the International Conference on Databases: Improving Usability and Responsiveness, Technion, Haifa, Israel.
- Gregor, S. (2006). The nature of theory in information systems. *MIS Quarterly*, 30(3), 611-642.
- Grinter, R. E., Herbsleb, J. D., & Perry, D. E. (1999, November 14-17, 1999). *The geography of coordination: dealing with distance in R&D work*. presented at the meeting of the Proceedings of the international ACM SIGGROUP conference on Supporting Group Work, Phoenix, Arizona, USA. doi:10.1145/320297.320333.
- Groth, D. (2005). An Evaluation of a Rule-Based Language for Classification Queries. In D. Seipel, M. Hanus, U. Geske, & O. Bartenstein (Eds.), *Applications of Declarative Programming and Knowledge Management* (Vol. 3392, pp. 79-97): Springer Berlin Heidelberg. doi:10.1007/11415763_6.
- Guetzkow, H., & Simon, H. A. (1955). The Impact of Certain Communication Nets upon Organization and Performance in Task-Oriented Groups. *Management Science*, 1(3/4), 233-250. doi:10.2307/2627162.
- Gummer, B. (2001). I'm in the Mood for Work -- Current Perspectives on Work Group Dynamics. *Administration in Social Work*, 25(2), 81 - 101.
- Gunawardena, C. N., Lowe, C., & Anderson, T. (1997). Analysis of a global on-line debate and the development of an interaction analysis model for examining

- social construction of knowledge in computer conferencing. *Journal of Educational Computing Research*, 17(4), 395-429.
- Gunawardena, C. N., & Zittle, F. J. (1997). Social presence as a predictor of satisfaction within a computer-mediated conferencing environment. *American Journal of Distance Education*, 11(3), 8-26.
- Hackman, J. R. (1986). The design of work teams. In *The handbook of Organizational Behavior* (Lorsch, J. W ed., pp. 315-342). Eaglewood Cliffs, NJ: Prentice-Hall.
- Hackman, J. R. (1992). Group influences on individuals in organizations. In M. D. Dunnette & L. M. Hough (Eds.), *Handbook of industrial and organizational psychology (Vol. 3)*. Palo Alto: Consulting Psychologists Press.
- Hackman, J. R., Morris, C. G., & Leonard, B. (1975). Group Tasks, Group Interaction Process, and Group Performance Effectiveness: A Review and Proposed Integration. In *Advances in Experimental Social Psychology* (Vol. Volume 8, pp. 45-99): Academic Press. doi:10.1016/S0065-2601(08)60248-8.
- Hall, T., Jagielska, D., & Baddoo, N. (2007). Motivating developer performance to improve project outcomes in a high maturity organization. *Software Quality Control*, 15(4), 365-381. doi:10.1007/s11219-007-9028-1.
- Hall, T., Wilson, D., Rainer, A., & Jagielska, D. (2007, April 19-21, 2007). *Communication: the neglected technical skill?* presented at the meeting of the Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research: The global information technology workforce, St. Louis, Missouri, USA. doi:10.1145/1235000.1235043.
- Han, J., & Kamber, M. (2006). *Data Mining: Concepts and Techniques* (2nd ed.). San Francisco: Morgan Kaufmann Publishers, Elsevier.
- Hancock, J. T., & Dunham, P. J. (2001). Impression Formation in Computer-Mediated Communication Revisited: An Analysis of the Breadth and Intensity of Impressions. *Communication Research*, 28(3), 325-347. doi:10.1177/009365001028003004.
- Hannay, J. E., Sjoberg, D. I. K., & Dyba, T. (2007). A Systematic Review of Theory Use in Software Engineering Experiments. *IEEE Transaction on Software Engineering*, 33(2), 87-107. doi:10.1109/tse.2007.12.
- Hanneman, R. A., & Riddle, M. (2005). *Introduction to social network methods*. Riverside, CA: University of California, Riverside.
- Hansen, D., Shneiderman, B., & Smith, M. (2011). *Analyzing Social Media Networks with NodeXL*. Boston: Morgan Kaufmann. doi:10.1016/b978-0-12-382229-1.00016-3.
- Hara, N., Bonk, C. J., & Angeli, C. (2000). Content analysis of online discussion in an applied educational psychology course. *Instructional Science*, 28(2), 115-152. doi:10.1023/a:1003764722829.
- Hart, R. P. (1984). *Verbal Style and the Presidency: A Computer-Based Analysis* New York: Academic Press.
- Hatano, G., & Inagaki, K. (1991). Sharing cognition through collective comprehension activity. In *In L.B. Resnick, J.M Levine, & S.D. Teasley (Eds.), Perspectives on socially shared cognition* (pp. 331-348). Washington DC: American Psychological Association.

- Hayes Huffman, J. (2003). Do you like Pina Coladas? How improved communication can improve software quality. *IEEE Software*, 20(1), 90-92.
- Heijstek, W., Kuhne, T., & Chaudron, M. R. V. (2011, September 22-23, 2011). *Experimental Analysis of Textual and Graphical Representations for Software Architecture Design*. presented at the meeting of the International Symposium on Empirical Software Engineering and Measurement (ESEM2011), Banff, AB. doi:10.1109/esem.2011.25.
- Hellriegel, D., & Slocum, J. W. (2007). *Organizational Behavior* (11 ed.). Mason, OH: Thomson Learning.
- Henri, F., & Kaye, A. R. (1992). Computer conferencing and content analysis. In *Collaborative learning through computer conferencing: The Najaden papers* (pp. 117-136). New York: Springer-Verlag.
- Henry, N., & Fekete, J.-D. (2007, September 10-14, 2007). *MatLink: enhanced matrix visualization for analyzing social networks*. presented at the meeting of the Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction - Volume Part II, Rio de Janeiro, Brazil.
- Herbsleb, J. D., & Grinter, R. E. (1999). Architectures, Coordination, and Distance: Conway's Law and Beyond. *IEEE Software*, 16(5), 63-70. doi:10.1109/52.795103.
- Herbsleb, J. D., & Mockus, A. (2003a). An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 29(6), 481-494. doi:10.1109/tse.2003.1205177.
- Herbsleb, J. D., & Mockus, A. (2003b, September 1-5, 2003). *Formulation and preliminary test of an empirical theory of coordination in software engineering*. presented at the meeting of the Proceedings of the 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Helsinki, Finland. doi:10.1145/940071.940091.
- Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2000, December 2-6, 2000). *Distance, dependencies, and delay in a global collaboration*. presented at the meeting of the Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, Philadelphia, Pennsylvania, United States. doi:10.1145/358916.359003.
- Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2001, May 12-19, 2001). *An empirical study of global software development: distance and speed*. presented at the meeting of the 23rd International Conference on Software Engineering, Toronto, Ontario, Canada. doi:10.1109/ICSE.2001.919083.
- Herbsleb, J. D., & Moitra, D. (2001). Global software development. *IEEE Software*, 18(2), 16-20.
- Herbsleb, J. D., & Roberts, J. (2006, December 10-13, 2006). *Collaboration In Software Engineering Projects: A Theory of Coordination*. presented at the meeting of the Proceedings of the International Conference on Information Systems, ICIS 2006, Milwaukee, Wisconsin, USA. doi:<http://aisel.aisnet.org/icis2006/38>.
- Herzig, K., & Zeller, A. (2009, May 16-17, 2009). *Mining the Jazz repository: Challenges and opportunities*. presented at the meeting of the 6th IEEE

- International Working Conference on Mining Software Repositories, 2009. MSR '09, Vancouver, BC. doi:10.1109/MSR.2009.5069495.
- Hevner, A., March, T., S, Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75-105.
- Highsmith, J. (2000). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. New York: Dorset House Publishing.
- Highsmith, J. (2004). *Agile Project Management: Creating Innovative Products*. Boston, MA: Pearson Education, Inc.
- Hinds, P. J., & McGrath, C. (2006, November 4 -8, 2006). *Structures that work: social structure, work structure and coordination ease in geographically distributed teams*. presented at the meeting of the Proceedings of the 2006 20th anniversary Conference on Computer Supported Cooperative Work, Banff, Alberta, Canada. doi:10.1145/1180875.1180928.
- Hinds, P. J., & Pfeffer, J. (2003). Why Organizations Don't Know What They Know: Cognitive and Motivational Factors Affecting the Transfer of Expertise. In *Ackerman, M, Pipek, V & Wulf, V, (Eds), Beyond Knowledge Management: Sharing Expertise* (citeulike:229884, pp. 3-26). Cambridge, MA: MIT Press.
- Hoda, R., Noble, J., & Marshall, S. (2010a, May 2, 2010). *Balancing acts: walking the Agile tightrope*. presented at the meeting of the Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, Cape Town, South Africa. doi:10.1145/1833310.1833312.
- Hoda, R., Noble, J., & Marshall, S. (2010b, May 1-8, 2010). *Organizing self-organizing teams*. presented at the meeting of the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, Cape Town, South Africa. doi:10.1145/1806799.1806843.
- Holden, R. R., & Passey, J. (2010). Socially desirable responding in personality assessment: Not necessarily faking and not necessarily substance. *Personality and Individual Differences*, 49(5), 446-450. doi:10.1016/j.paid.2010.04.015.
- Holsti, O. R. (1969). *Content Analysis for the Social Sciences and Humanities*. Reading, MA: Addison Wesley.
- Howe, K. R. (1988). Against the Quantitative-Qualitative Incompatibility Thesis or Dogmas Die Hard. *Educational Researcher*, 17(8), 10-16. doi:10.3102/0013189x017008010.
- Howison, J., Inoue, K., & Crowston, K. (2006). Social dynamics of free and open source team communications. In *Open Source Systems* (Vol. 203, pp. 319-330): Springer Boston. doi:10.1007/0-387-34226-5_32.
- Hsieh, H.-F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, 15(9), 1277-1288. doi:10.1177/1049732305276687.
- Humphrey, W. S. (1997). *Introduction to the personal software process*. Boston, MA: Addison-Wesley Longman Publishing Co.
- Humphrey, W. S. (1998). Three Dimensions of Process Improvement. Part III: The Team Process. *CROSSTALK The Journal of Defense Software Engineering*, 11(2), 14-17.

- Hussey, J., & Hussey, R. (1997). *Business Research: A Practical Guide for Undergraduate and Postgraduate Students*. London: Macmillan.
- Inkpen, A. C., & Tsang, E. W. K. (2005). Social Capital, Networks, and Knowledge Transfer. *The Academy of Management Review*, 30(1), 146-165. doi:10.2307/20159100.
- Ivancevich, J. M., & Matteson, M. T. (2001). *Organizational Behavior and Management*. (6 th Edition ed.): McGraw-hill Professional.
- Jaanu, T., Paasivaara, M., & Lassenius, C. (2012, September 19-20, 2012). *Effects of four distances on communication processes in global software projects*. presented at the meeting of the Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, Lund, Sweden. doi:10.1145/2372251.2372293.
- Jalali, S., & Wohlin, C. (2010, August 23-26, 2010). *Agile Practices in Global Software Engineering - A Systematic Map*. presented at the meeting of the 2010 5th IEEE International Conference on Global Software Engineering (ICGSE), Princeton, NJ. doi:10.1109/icgse.2010.14.
- Jamali, M., & Abolhassani, H. (2006, December 18-22, 2006). *Different Aspects of Social Network Analysis*. presented at the meeting of the Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, Hong Kong. doi:10.1109/wi.2006.61.
- James, J. (1951). A Preliminary Study of the Size Determinant in Small Group Interaction. *American Sociological Review*, 16(4), 474-477. doi:10.2307/2088278.
- Johansen, R. (1977). Social evaluations of teleconferencing. *Telecommunications Policy*, 1(5), 395-419.
- Johnson, P., Ekstedt, M., & Jacobson, I. (2012). Where's the Theory for Software Engineering? *IEEE Software* 29(5), 96-96. doi:10.1109/ms.2012.127.
- Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed Methods Research: A Research Paradigm Whose Time Has Come. *Educational Researcher*, 33(7), 14-26. doi:10.3102/0013189x033007014.
- Jonassen, D., & Kwon, H. (2001). Communication patterns in computer mediated versus face-to-face group problem solving. *Educational Technology Research and Development*, 49(1), 35-51. doi:10.1007/bf02504505.
- Jones, C. (1996). Strategies for managing requirements creep. *Computer*, 29(6), 92-94. doi:10.1109/2.507640.
- Jung, C. (1971). *Psychological types*. (Vol. 6). New Jersey: Princeton University Press.
- Junior, M. C., Mendonca, M., Farias, M., & Henrique, P. (2010, May 2-3, 2010). *OSS developers context-specific Preferred Representational systems: A initial Neurolinguistic text analysis of the Apache mailing list*. presented at the meeting of the 7th IEEE Working Conference on Mining Software Repositories (MSR), Cape Town, SA. doi:10.1109/MSR.2010.5463339.
- Kalman, M. E., Monge, P., Fulk, J., & Heino, R. (2002). Motivations to Resolve Communication Dilemmas in Database-Mediated Collaboration. *Communication Research*, 29(2), 125-154. doi:10.1177/0093650202029002002.

- Kamaruddin, N. K., Arshad, N. H., & Mohamed, A. (2012, April 7-8, 2012). *Chaos issues on communication in Agile Global Software Development*. presented at the meeting of the 2012 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC), Kuala Lumpur. doi:10.1109/beiac.2012.6226091.
- Kampenès, V. B., Dybå, T., Hannay, J. E., & Sjøberg, D. I. K. (2007). A systematic review of effect size in software engineering experiments. *Information and Software Technology*, 49(11–12), 1073-1086. doi:10.1016/j.infsof.2007.02.015.
- Kanawattanachai, P., & Yoo, Y. (2007). The Impact of Knowledge Coordination on Virtual Team Performance over Time. *MIS Quarterly*, 31(4), 783-808. doi:10.2307/25148820.
- Kankanhalli, A., Tan, B., & Wei, K.-K. (2005). Contributing knowledge to electronic knowledge repositories: an empirical investigation. *MIS Quarterly*, 29(1), 113-143.
- Karn, J., & Cowling, T. (2006, September 21-22, 2006). *A follow up study of the effect of personality on the performance of software engineering teams*. presented at the meeting of the Proceedings of the 2006 ACM/IEEE international symposium on Empirical Software Engineering, Rio de Janeiro, Brazil. doi:10.1145/1159733.1159769.
- Karolák, D. W. (1999). *Global Software Development: Managing Virtual Teams and Environments*. Los Alamitos, CA: IEEE Computer Society Press.
- Katzenbach, J., & Smith, D. (2001). *The Discipline of Teams: A Mindbook-Workbook for Delivering Small Group Performance* (First Edition ed.). New York: John Wiley & Sons.
- Kennedy, D. M., McComb, S. A., & Vozdolska, R. R. (2011). An investigation of project complexity's influence on team communication using Monte Carlo simulation. *Journal of Engineering and Technology Management*, 28(3), 109-127. doi:10.1016/j.jengtecman.2011.03.001.
- Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1), 67-93. doi:10.2307/249410.
- Kline, C. J., & Peters, L. H. (1991). Behavioral Commitment and Tenure of New Employees: A Replication and Extension. *The Academy of Management Journal*, 34(1), 194-204. doi:10.2307/256307.
- Koch, A. (2005). *Agile Software Development: Evaluating the methods for your organisation*. Boston, MA: Artech House.
- Krauss, R. M., & Fussell, S. R. (1996). Social psychological models of interpersonal communication. In E. T. Higgins & A. Kruglanski (Eds.), *Social psychology: Handbook of basic principles* (pp. 655-701): Guilford Press.
- Krebs, S. A., Hobman, E. V., & Bordia, P. (2006). Virtual Teams and Group Member Dissimilarity: Consequences for the Development of Trust. *Small Group Research*, 37(6), 721-741. doi:10.1177/1046496406294886.
- Krippendorff, K. (2004). *Content Analysis: An Introduction to Its Methodology* (2nd ed.). Thousand Oaks, CA: Sage Publications.
- Kuhn, T. (1970). *The structure of scientific revolutions*. (Second ed.). Chicago: The University of Chicago Press.

- Kuzel, A. (1992). Sampling in qualitative inquiry. In *Doing qualitative research* (B. Crabtree and W. Miller ed., pp. 31 - 44). Newbury Park, CA: Sage.
- Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development: A Brief History. *Computer*, 36(6), 47-56. doi:10.1109/mc.2003.1204375.
- Larose, D. T. (2005). *Discovering Knowledge in Data: An introduction to Data Mining*. Hoboken, NJ: John Wiley & Sons, Inc. doi:10.1002/0471687545.
- Layman, L., Williams, L., Damian, D., & Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. *Information and Software Technology*, 48(9), 781-794. doi:10.1016/j.infsof.2006.01.004.
- Lázaro, M., & Marcos, E. (2006, June 5-9, 2006). *An Approach to the Integration of Qualitative and Quantitative Research Methods in Software Engineering Research*. presented at the meeting of the 2nd CAiSE International Workshop on Philosophical Foundations of Information Systems Engineering (PHISE'06), Luxembourg.
- Leavitt, H. J. (1951). Some effects of certain communication patterns on group performance. *The Journal of Abnormal & Social Psychology*, 46(1), 38-50.
- Lee, D. M. S., Trauth, E. M., & Farwell, D. (1995). Critical skills and knowledge requirements of IS professionals: a joint academic/industry investigation. *MIS Quarterly*, 19(3), 313-340. doi:10.2307/249598.
- Lee, G., & Xia, W. (2005). The ability of information systems development project teams to respond to business and technology changes: a study of flexibility measures. *European Journal of Information Systems*, 14(1), 75-92.
- Lee, S., & Yong, H.-S. (2010). Distributed agile: project management in a global environment. *Empirical Software Engineering*, 15(2), 204-217. doi:10.1007/s10664-009-9119-7.
- Leech, N., & Onwuegbuzie, A. (2009). A typology of mixed methods research designs. *Quality & Quantity*, 43(2), 265-275. doi:10.1007/s11135-007-9105-3.
- Levin, D. Z., & Cross, R. (2004). The Strength of Weak Ties You Can Trust: The Mediating Role of Trust in Effective Knowledge Transfer. *Management Science*, 50(11), 1477-1490. doi:10.2307/30047959.
- Lewin, K. (1945). The Research Center for Group Dynamics at Massachusetts Institute of Technology. *Sociometry*, 8(2), 126-136. doi:10.2307/2785233.
- Li, J., & Chignell, M. (2010). Birds of a feather: How personality influences blog writing and reading. *International Journal of Human-Computer Studies*, 68(9), 589-602.
- Li, Z., Tan, L., Wang, X., Lu, S., Zhou, Y., & Zhai, C. (2006, October 21, 2006). *Have things changed now?: an empirical study of bug characteristics in modern open source software*. presented at the meeting of the Proceedings of the 1st workshop on Architectural and system support for improving software dependability, San Jose, California. doi:10.1145/1181309.1181314.
- Licorish, S. A., & MacDonell, S. G. (2012, June 2, 2012). *What affects team behavior? Preliminary linguistic analysis of communications in the Jazz repository*. presented at the meeting of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2012), Zurich, Switzerland. doi:10.1109/chase.2012.6223029.

- Licorish, S. A., & MacDonell, S. G. (2013a, April 14-16, 2013). *Adopting Softer Approaches in the Study of Repository Data: A Comparative Analysis*. presented at the meeting of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE 2013), Porto de Galinhas, Brazil. doi:10.1145/2460999.2461035.
- Licorish, S. A., & MacDonell, S. G. (2013b, May 25, 2013). *Differences in Jazz project leaders' competencies and behaviors: a preliminary empirical investigation*. presented at the meeting of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2013), San Francisco, CA.
- Licorish, S. A., & MacDonell, S. G. (2013c, April 14-16, 2013). *The true role of active communicators: an empirical study of Jazz core developers*. presented at the meeting of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE 2013), Porto de Galinhas, Brazil. doi:10.1145/2460999.2461034.
- Licorish, S. A., & MacDonell, S. G. (2013d, June 4-7, 2013). *What can developers' messages tell us?: A psycholinguistic analysis of Jazz teams' attitudes and behavior patterns*. presented at the meeting of the 22th Australian Conference on Software Engineering (ASWEC 2013), Melbourne, Australia. doi:10.1109/ASWEC.2013.22.
- Licorish, S. A., Philpott, A., & MacDonell, S. G. (2009a, July 13-16, 2009). *A prototype tool to support extended team collaboration in agile project feature management*. presented at the meeting of the International Conference on Software Engineering Theory and Practice (SETP 2009), Orlando FL, USA.
- Licorish, S. A., Philpott, A., & MacDonell, S. G. (2009b, May 17, 2009). *Supporting agile team composition: A prototype tool for identifying personality (In)compatibilities*. presented at the meeting of the ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE 2009), Vancouver, Canada. doi:10.1109/CHASE.2009.5071413.
- Litecky, C. R., Arnett, K. P., & Prabhakar, B. (2004). The paradox of soft skills versus technical skills in IS hiring. *The Journal of Computer Information Systems*, 45(1), 69-76.
- Ljungberg, J. (2000). Open source movements as a model for organising. *European Journal of Information Systems*, 9(4), 208-216. doi:10.1057/palgrave.ejis.3000373.
- Lloyd, K. B., & Jankowski, D. J. (1999). A cognitive information processing and information theory approach to diagram clarity: a synthesis and experimental investigation. *Journal of Systems and Software*, 45(3), 203-214. doi:10.1016/s0164-1212(98)10079-1.
- Lockhorst, D., Admiraal, W., Pilot, A., & Veen, W. (2003). *Analysis of electronic communication using 5 different perspectives*. Heerlen, Netherlands.
- Lynham, S. A. (2002). The General Method of Theory-Building Research in Applied Disciplines. *Advances in Developing Human Resources*, 4(3), 221-241. doi:10.1177/1523422302043002.
- Mairesse, F., & Walker, M. (2006, June 4-9, 2006). *Automatic recognition of personality in conversation*. presented at the meeting of the Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, New York.

- Mairesse, F., Walker, M., Mehl, M. R., & Moore, R. K. (2007). Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research*, 30(1), 457-500.
- Manca, S., Delfino, M., & Mazzoni, E. (2009). Coding procedures to analyse interaction patterns in educational web forums. *Journal of Computer Assisted Learning*, 25(2), 189-200. doi:10.1111/j.1365-2729.2008.00296.x.
- Margerison, C. J., McCann, D. J., & Davies, R. (1986). The Margerison-McCann Team Management Resource - Theory and Applications. *International Journal of Manpower*, 7(2), 3-32.
- Markus, M. L., Manville, B., & Agres, E. C. (2000). What makes a virtual organization work? *Sloan Management Review*, 42(1), 13-26.
- Mayring, P. (2000). Qualitative content analysis. *Forum: Qualitative Social Research*, 1(2). Advance online publication. 471 Retrieved from <http://www.qualitative-research.net/index.php/fqs/article/view/1089>
- McCrae, R. R., & Costa, P. T. (1987). Validation of the Five-Factor Model of Personality Across Instruments and Observers. *Journal of Personality & Social Psychology*, 52(1), 81-90.
- McDonough, E. F., Kahn, K. B., & Barczak, G. (2001). An investigation of the use of global, virtual, and colocated new product development teams. *Journal of Product Innovation Management*, 18(2), 110-120. doi:10.1016/S0737-6782(00)00073-4.
- McGann, S., & Lyytinen, K. (2008, December 14-17, 2008). *The improvisation effect: a case study of user Improvisation and its effects on information system evolution*. presented at the meeting of the Proceedings of the 29th International Conference on Information Systems, ICIS 2008, Paris, France.
- Mehl, M. R., & Pennebaker, J. W. (2003). The sounds of social life: a psychometric analysis of students' daily social environments and natural conversations. *Journal of Personality and Social Psychology*, 84(4), 857-870.
- Mendieta, J. G., Schmidt, S., & Ruiz, A. (1997). A Dynamic Analysis of the Mexican Power Network. *Connections*, 20(2), 34-55.
- Messick, S. (1989). Validity. . In R.L. Linn (Ed.), in *Educational Measurement* (3rd ed., pp. 13-103). New York: American Council on Education & Macmillan.
- Miles, M., & Huberman, A. (1994). *Qualitative data analysis*. Thousand Oaks, CA: Sage.
- Miller, J., Daly, J., Wood, M., Roper, M., & Brooks, A. (1997). Statistical power and its subcomponents — missing and misunderstood concepts in empirical software engineering research. *Information and Software Technology*, 39(4), 285-295. doi:10.1016/S0950-5849(96)01139-1.
- Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3), 309-346. doi:10.1145/567793.567795.
- Mockus, A., & Votta, L. G. (2000, October 11-14, 2000). *Identifying Reasons for Software Changes Using Historic Databases*. presented at the meeting of the Proceedings of the International Conference on Software Maintenance (ICSM'00), San Jose, CA. doi:10.1109/ICSM.2000.883028.

- Moe, N. B., Dingsoyr, T., & Dyba, T. (2008, March 26-28, 2008). *Understanding Self-Organizing Teams in Agile Software Development*. presented at the meeting of the Proceedings of the 19th Australian Conference on Software Engineering, Perth, WA. doi:10.1109/ASWEC.2008.4483195.
- Moe, N. B., Dingsoyr, T., & Dyba, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology*, 52(5), 480-491.
- Moreno, J. L. (1953). *Who Shall Survive?* Beacon, New York: Beacon House, Inc.
- Morgan, R. M., & Hunt, S. D. (1994). The Commitment-Trust Theory of Relationship Marketing. *Journal of Marketing*, 58(3), 20-38. doi:10.2307/1252308.
- Morrison, J., & George, J. F. (1995). Exploring the software engineering component in MIS research. *Communications of the ACM*, 38(7), 80-91. doi:10.1145/213859.214802.
- Mowday, R. T., Steers, R. M., & Porter, L. W. (1979). The measurement of organizational commitment. *Journal of Vocational Behavior*, 14(2), 224-247. doi:10.1016/0001-8791(79)90072-1.
- Mowrer, D. E. (1996). A content analysis of student/instructor communication via computer conferencing. *Higher Education*, 32(2), 217-241. doi:10.1007/bf00138397.
- Mulac, A., Bradac, J. J., & Gibbons, P. (2001). Empirical support for the gender-as-culture hypothesis. *Human Communication Research*, 27(1), 121-152. doi:10.1111/j.1468-2958.2001.tb00778.x.
- Mumford, M. D., & Gustafson, S. B. (1988). Creativity Syndrome: Integration, Application, and Innovation. *Psychological Bulletin*, 103(1), 27-43.
- Nagappan, N., & Ball, T. (2007, September 20-21, 2007). *Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study*. presented at the meeting of the First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, Spain. doi:10.1109/esem.2007.13.
- Nagappan, N., Murphy, B., & Basili, V. (2008, May 10-18, 2008). *The influence of organizational structure on software quality: an empirical case study*. presented at the meeting of the Proceedings of the 30th international conference on Software engineering, Leipzig, Germany. doi:10.1145/1368088.1368160.
- Naidu, S., & Jarvela, S. (2006). Analyzing CMC content for what? *Computers & Education*, 46(1), 96-103. doi:10.1016/j.compedu.2005.04.001.
- Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5), 72 - 78. doi:10.1145/1060710.1060712.
- Neuendorf, K. A. (2002). *The content analysis guidebook*. Thousand Oaks: Sage Publication.
- Newman, D. R., Webb, B., & Cochrane, C. (1995). A Content Analysis Method to Measure Critical Thinking in Face-to-Face and Computer Supported Group Learning. *Interpersonal Computing and Technology*, 3(2), 56-77.

- Newman, I., & Benz, C. R. (1998). *Qualitative-quantitative research methodology: Exploring the interactive continuum*. Carbondale: Southern Illinois University Press.
- Nguyen, T., Schroter, A., & Damian, D. (2008, November 9, 2008). *Mining Jazz: An experience report*. presented at the meeting of the In Proceedings of the 1st International Workshop on Infrastructure for Research in Collaborative Software Engineering(iReCoSE), Atlanta, Georgia, USA.
- Nguyen, T., Wolf, T., & Damian, D. (2008, August 17-20, 2008). *Global Software Development and Delay: Does Distance Still Matter?* presented at the meeting of the IEEE International Conference on Global Software Engineering (ICGSE 2008), Bangalore, India. doi:10.1109/ICGSE.2008.39.
- Nord, R. L., & Tomayko, J. E. (2006). Software Architecture-Centric Methods and Agile Development. *IEEE Software*, 23(2), 47-54.
- Nowson, S., & Oberlander, J. (2006). *The Language of Weblogs: A study of genre and individual differences*. University of Edinburgh. College of Science and Engineering.
- O'Dell, C., & Grayson, C. J. (1998). If only we knew what we know: Identification and transfer of internal best practices. *California Management Review*, 40(3), 154-174.
- Ocker, R., J., & Fjermestad, J. (2008). Communication differences in virtual design teams: findings from a multi-method analysis of high and low performing experimental teams. *SIGMIS Database*, 39(1), 51-67. doi:10.1145/1341971.1341977.
- OED-Online. *Oxford English Dictionary Online* (539). Retrieved February 1, 2013, from <http://www.oed.com/>
- Oh, H., Labianca, G., & Chung, M. (2006). A Multilevel Model of Group Social Capital. *Academy of Management Review*, 31(3), 569-582.
- Oh, W., Gallivan, M. J., & Kim, J. W. (2006). The Market's Perception of the Transactional Risks of Information Technology Outsourcing Announcements. *Journal of Management Information Systems*, 22(4), 271-303. doi:10.2307/40398820.
- Onwuegbuzie, A. J. (2003). Effect Sizes in Qualitative Research: A Prolegomenon. *Quality & Quantity*, 37(4), 393-409. doi:10.1023/a:1027379223537.
- Onwuegbuzie, A. J., & Danlel, L. G. (2002). Uses and misuses of correlation coefficient. *Research in the Schools*, 9(1), 73-90.
- Onwuegbuzie, A. J., & Leech, N. (2005). On Becoming a Pragmatic Researcher: The Importance of Combining Quantitative and Qualitative Research Methodologies. *International Journal of Social Research Methodology*, 8(5), 375-387. doi:10.1080/13645570500402447.
- Onwuegbuzie, A. J., & Teddlie, C. (2003). A framework for analyzing data in mixed methods research. In *In A Tashakkori & C Teddlie (Eds.), Handbook of mixed methods in social and behavioral research* (pp. 351-383). Thousand Oaks, CA: Sage.
- Oreg, S., & Nov, O. (2008). Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. *Computers in Human Behavior*, 24(5), 2055-2073. doi:10.1016/j.chb.2007.09.007.

- Orlikowski, W. (1992). The Duality of Technology: Rethinking the Concept of Technology in Organizations. *ORGANIZATION SCIENCE*, 3(3), 398-427.
- Oxman, T. E., Rosenberg, S. D., Schnurr, P. P., & Tucker, G. J. (1988). Diagnostic classification through content analysis of patients' speech. *American Journal of Psychiatry*, 145(4), 464-468.
- Paasivaara, M., & Lassenius, C. (2003). Collaboration practices in global inter-organizational software development projects. *Software Process: Improvement and Practice*, 8(4), 183-199. doi:10.1002/spip.187.
- Palazzolo, E. T., Serb, D. A., She, Y., Su, C., & Contractor, N. S. (2006). Coevolution of Communication and Knowledge Networks in Transactive Memory Systems: Using Computational Models for Theoretical Development. *Communication Theory*, 16(2), 223-250. doi:10.1111/j.1468-2885.2006.00269.x.
- Pattison, D. S., Bird, C. A., & Devanbu, P. T. (2008, May 10-18, 2008). *Talk and work: a preliminary report*. presented at the meeting of the Proceedings of the 2008 international working conference on Mining software repositories, Leipzig, Germany. doi:10.1145/1370750.1370776.
- Paul, L. B. (2008). Risk and risk management in software projects: A reassessment. *Journal of Systems and Software*, 81(12), 2118-2133. doi:10.1016/j.jss.2008.03.059.
- Pennebaker, J., Chung, C., Ireland, M., Gonzales, A., & Booth, R. J. (2007). *Linguistic inquiry and word count* (291). Retrieved 10-01-2011, from <http://liwc.net/index.php>
- Pennebaker, J., & King, L. (1999). Linguistic Styles: Language Use as an Individual Difference. *Journal of Personality & Social Psychology*, 77(6), 1296-1312. doi:10.1037//0022-3514.77.6.1296.
- Pennebaker, J., & Lay, T. (2002). Language Use and Personality during Crises: Analyses of Mayor Rudolph Giuliani's Press Conferences. *Journal of Research in Personality*, 36(3), 271-282. doi:10.1006/jrpe.2002.2349.
- Pennebaker, J., Mayne, T., & Francis, M. (1997). Linguistic Predictors of Adaptive Bereavement. *Journal of Personality & Social Psychology*, 72(4), 863-871.
- Pennebaker, J., Mehl, M., & Niederhoffer, K. (2003). Psychological Aspects of Natural Language Use: Our Words, Our Selves. *Annual Review of Psychology*, 54(1), 547-577. doi:10.1146/annurev.psych.54.101601.145041.
- Pennebaker, J., & Stone, L. (2003). Words of Wisdom: Language Use Over the Life Span. *Journal of Personality and Social Psychology*, 85(2), 291-301. doi:10.1037/0022-3514.85.2.29.
- Perry, D. E., Sim, S. E., & Easterbrook, S. M. (2004, May 23-28, 2004). *Case studies for software engineers*. presented at the meeting of the 26th International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, UK. doi:10.1109/icse.2004.1317512.
- Perry, D. E., Staudenmayer, N., & Votta, L. G. (1994). People, Organizations, and Process Improvement. *IEEE Software*, 11(4), 36-45. doi:10.1109/52.300082.
- Pfleeger, S. L. (1999). Albert Einstein and empirical software engineering. *Computer*, 32(10), 32-38. doi:10.1109/2.796106.

- Pinzger, M., Nagappan, N., & Murphy, B. (2008, November 9-14, 2008). *Can developer-module networks predict failures?* presented at the meeting of the Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, Atlanta, Georgia. doi:10.1145/1453101.1453105.
- Pohl, M., & Diehl, S. (2008, May 13, 2008). *What dynamic network metrics can tell us about developer roles.* presented at the meeting of the Proceedings of the 2008 International workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2008), Leipzig, Germany. doi:10.1145/1370114.1370135.
- Pollock, M. (2009). Investigating the relationship between team role diversity and team performance in information systems teams. *Journal of Information Technology Management*, 20(1), 42-55.
- Polzehl, T., Moller, S., & Metze, F. (2010, September 22-24, 2010). *Automatically Assessing Personality from Speech.* presented at the meeting of the IEEE Fourth International Conference on Semantic Computing (ICSC), 2010 Pittsburgh, PA. doi:10.1109/ICSC.2010.41.
- Potts, C. (1993). Software-engineering research revisited. *IEEE Software*, 10(5), 19-28. doi:10.1109/52.232392.
- Powell, W. W. (1990). Neither market nor hierarchy: network forms of organization. In B. M. Staw and L. L. Cummins, eds. *Research in Organizational Behavior*, 12, 295-336.
- Prabhala, S., & Gallimore, J. J. (2005, May 14-8, 2005). *Can humans perceive personality in computer agents?* presented at the meeting of the IIE Annual Conference and Exposition 2005, Atlanta Georgia.
- Prashant, B. (1997). Face-to-Face Versus Computer-Mediated Communication: A Synthesis of the Experimental Literature. *Journal of Business Communication*, 34(1), 99 -120.
- Pressman, R. S. (2009). *Software Engineering: A Practitioner's Approach* (7th ed.). New York: McGraw-Hill.
- Prikladnicki, R., Nicolas Audy, J. L., & Evaristo, R. (2003). Global software development in practice lessons learned. *Software Process: Improvement and Practice*, 8(4), 267-281. doi:10.1002/spip.188.
- Proper, K. (2005). *The logic of scientific discovery*. London: Taylor and Francis Group.
- Quigley, N. R., Tesluk, P. E., Locke, E. A., & Bartol, K. M. (2007). A Multilevel Investigation of the Motivational Mechanisms Underlying Knowledge Sharing and Performance. *Organization Science*, 18(1), 71-88. doi:10.1287/orsc.1060.0223.
- Rajendran, M. (2005). Analysis of team effectiveness in software development teams working on hardware and software environments using Belbin Self-Perception inventory. *Journal of Management Development*, 24(8), 738-753. doi:10.1108/02621710510613753.
- Ramesh, V., Glass, R. L., & Vessey, I. (2004). Research in computer science: An empirical study. *Journal of Systems and Software*, 70(1-2), 165-176. doi:10.1016/S0164-1212(03)00015-3.
- Ravenscroft, A., & Pilkington, R. M. (2000). Investigation by Design: Developing Dialogue Models to Support Reasoning and Conceptual Change. *International*

Journal of Artificial Intelligence in Education: Special Issue on Analysing Educational Dialogue Interaction: From Analysis to Models that Support Learning, 11(3), 273--298.

- Reagans, R., & Zuckerman, E. W. (2001). Networks, Diversity, and Productivity: The Social Capital of Corporate R&D Teams. *Journal of Organization Science*, 12(4), 502-517. doi:10.1287/orsc.12.4.502.10637.
- Reid, A. (1977). *Comparing telephone with face-to-face contact*. Cambridge, Mass: MIT Press.
- Rice, R. E. (1992). Task Analyzability, Use of New Media, and Effectiveness: A Multi-Site Exploration of Media Richness. *Organization Science*, 3(4), 475 - 500.
- Rich, S. (2010, April 26-30, 2010). *IBM's jazz integration architecture: building a tools integration architecture and community inspired by the web*. presented at the meeting of the Proceedings of the 19th international conference on World wide web, Raleigh, North Carolina, USA. doi:10.1145/1772690.1772936.
- Richardson, J., & Swan, K. (2003). Examining Social Presence in Online Courses in Relation to Student's Perceived Learning and Satisfaction. *Journal of Asynchronous Learning Networks*, 7(1), 68- 88.
- Ricoeur, P. (1981). *Hermeneutics and the Human Sciences*. Cambridge, UK: Cambridge University Press.
- Rigby, P., & Hassan, A. (2007, May 20-26, 2007). *What Can OSS Mailing Lists Tell Us? A Preliminary Psychometric Text Analysis of the Apache Developer Mailing List*. presented at the meeting of the Proceedings of the Fourth International Workshop on Mining Software Repositories, Minneapolis, MN. doi:10.1109/msr.2007.35.
- Rigby, P., & Storey, M.-A. (2011, May 21-28, 2011). *Understanding broadcast based peer review on open source software projects*. presented at the meeting of the 2011 33rd International Conference on Software Engineering (ICSE), Honolulu, HI. doi:10.1145/1985793.1985867.
- Robles, G., Gonzalez-Barahona, J. M., & Herraiz, I. (2009, May 16-17, 2009). *Evolution of the core team of developers in libre software projects*. presented at the meeting of the 6th IEEE International Working Conference on Mining Software Repositories, 2009 (MSR '09), Vancouver, BC. doi:10.1109/msr.2009.5069497.
- Rodriguez, D., Herraiz, I., & Harrison, R. (2012, June 5, 2012). *On software engineering repositories and their open problems*. presented at the meeting of the First International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012, Zurich, Switzerland. doi:10.1109/raise.2012.6227971.
- Romney, A. K., Weller, S. C., & Batchelder, W. H. (1986). Culture as Consensus: A Theory of Culture and Informant Accuracy. *American Anthropologist*, 88(2), 313-338. doi:10.2307/677564.
- Ropponen, J., & Lyytinen, K. (2000). Components of software development risk: how to address them? A project manager survey. *IEEE Transactions on Software Engineering*, 26(2), 98-112.

- Rosengren, K. E. (1981). Advances in Scandinavia content analysis: An introduction. In K. E. Rosengren (Ed.), *Advances in content analysis* (pp. 9-19). Beverly Hills, CA: Sage.
- Rourke, L., & Anderson, T. (2002). Exploring Social Communication in Computer Conferencing. *Journal of Interactive Learning Research*, 13(3), 259-275.
- Rourke, L., & Anderson, T. (2004). Validity in quantitative content analysis. *Educational Technology Research and Development*, 52(1), 5-18. doi:10.1007/bf02504769.
- Rourke, L., Anderson, T., Garrison, D. R., & Archer, W. (2001). Methodological issues in the content analysis of computer conference transcripts. *International Journal of Artificial Intelligence in Education*, 12(1), 8-22.
- Rowley, D., & Lange, M. (2007, August 13-17, 2007). *Forming to Performing: The Evolution of an Agile Team*. presented at the meeting of the Agile Conference (AGILE), 2007, Washington, DC. doi:10.1109/agile.2007.28.
- Rudzki, J., Hammouda, I., Mikkola, T., Mustonen, K., & Systä, T. (2010). Considering Subcontractors in Distributed Scrum Teams. In D. Šmite, N. B. Moe, & P. J. Ågerfalk (Eds.), *Agility Across Time and Space* (pp. 235-255): Springer Berlin Heidelberg. doi:10.1007/978-3-642-12442-6_16.
- Ruhnow, A. (2007, August 13-17, 2007). *Consciously Evolving an Agile Team*. presented at the meeting of the Agile Conference (AGILE), 2007, Washington, DC. doi:10.1109/agile.2007.20.
- Runeson, P., & Host, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 131-164. doi:10.1007/s10664-008-9102-8.
- Ryan, S., & O'Connor, R. V. (2009). Development of a team measure for tacit knowledge in software development teams. *Journal of Systems and Software*, 82(2), 229-240. doi:10.1016/j.jss.2008.05.037.
- Sach, R., Sharp, H., & Petre, M. (2011, September 19-23, 2011). *Software Engineers' Perceptions of Factors in Motivation*. presented at the meeting of the 5th International Symposium on Empirical Software Engineering and Measurement Banff, Alberta, Canada.
- Sahay, S. (2003). Global software alliances: the challenge of 'standardization'. *Scandinavian Journal of Information Systems*, 15(1), 3-21.
- Sauer, C., Jeffery, D. R., Land, L., & Yetton, P. (2000). The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research. *IEEE Trans. Softw. Eng.*, 26(1), 1-14. doi:10.1109/32.825763.
- Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying Software Project Risks: An International Delphi Study. *Journal of Management Information Systems*, 17(4), 5-36.
- Schnurr, P. P., Rosenberg, S. D., & Oxman, T. E. (1992). Comparison of TAT and Free Speech Techniques for Eliciting Source Material in Computerized Content Analysis. *Journal of Personality Assessment*, 58(2), 311-325. doi:10.1207/s15327752jpa5802_10.
- Schrire, S. (2006). Knowledge building in asynchronous discussion groups: Going beyond quantitative analysis. *Journal of Computers & Education*, 46(1), 49-70. doi:10.1016/j.compedu.2005.04.006.

- Schroter, A. (2010, May 2-8, 2010). *Predicting build outcome with developer interaction in Jazz*. presented at the meeting of the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, Cape Town, South Africa. doi:10.1145/1810295.1810456.
- Schultz, M., & Hatch, M. J. (1996). Living with Multiple Paradigms: The Case of Paradigm Interplay in Organizational Culture Studies. *The Academy of Management Review*, 21(2), 529-557. doi:10.2307/258671.
- Scott, J. (2000). *Social Network Analysis: A Handbook* (Scott00). London: Sage Publications.
- Serce, F. C., Alpaslan, F.-N., Swigger, K., Brazile, R., Dafoulas, G., Lopez, V. (2009, July 13-16, 2009). *Exploring Collaboration Patterns among Global Software Development Teams*. presented at the meeting of the Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering, Limerick, Ireland. doi:10.1109/icgse.2009.14.
- Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton: Mifflin.
- Sharma, V. S., & Kaulgud, V. (2011, May 21, 2011). *Studying team evolution during software testing*. presented at the meeting of the Proceeding of the 4th International workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2011), Waikiki, Honolulu, HI, USA. doi:10.1145/1984642.1984660.
- Sharp, H., Baddoo, N., Beecham, S., Hall, T., & Robinson, H. (2009). Models of motivation in software engineering. *Information and Software Technology*, 51(1), 219-233. doi:10.1016/j.infsof.2008.05.009.
- Sharp, V. (1979). *Statistics for the social sciences*: the University of Michigan.
- Sheetz, S. D., Henderson, D., & Wallace, L. (2009). Understanding developer and manager perceptions of function points and source lines of code. *Journal of Systems and Software*, 82(9), 1540-1549. doi:10.1016/j.jss.2009.04.038.
- Shihab, E., Bettenburg, N., Adams, B., & Hassan, A. (2010). On the Central Role of Mailing Lists in Open Source Projects: An Exploratory Study. In *New Frontiers in Artificial Intelligence* (Vol. 6284, pp. 91-103). Heidelberg: Springer Berlin. doi:10.1007/978-3-642-14888-0_9.
- Shihab, E., Zhen Ming, J., & Hassan, A. (2009, September 20-26, 2009). *Studying the use of developer IRC meetings in open source projects*. presented at the meeting of the IEEE International Conference on Software Maintenance (ICSM 2009), Edmonton, AB. doi:10.1109/ICSM.2009.5306333.
- Short, J., Williams, E., & Christie, B. (1976). *The Social Psychology of Telecommunications*. London, England: John Wiley & Sons Ltd.
- Siddiqui, M. S., & Hussain, S. J. (2006, March 8, 2006). *Comprehensive Software Development Model*. presented at the meeting of the IEEE International Conference on Computer Systems and Applications, Dubai/Sharjah, UAE. doi:10.1109/AICCSA.2006.205113.
- Singer, J. (1998, November 16-20, 1998). *Practices of Software Maintenance*. presented at the meeting of the Proceedings of the International Conference on Software Maintenance, Bethesda, MD. doi:10.1109/ICSM.1998.738502.

- Sinialto, M., & Abrahamsson, P. (2007, September 20-21, 2007). *A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage*. presented at the meeting of the First International Symposium on Empirical Software Engineering and Measurement, (ESEM 2007) Madrid, Spain. doi:10.1109/esem.2007.35.
- Smilowitz, M., Compton, D. C., & Flint, L. (1988). The effects of computer mediated communication on an individual's judgment: A study based on the methods of Asch's social influence experiment. *Computers in Human Behavior*, 4(4), 311 - 321.
- Smith, E., Seger, C., & Mackie, D. (2007). Can Emotions Be Truly Group Level? Evidence Regarding Four Conceptual Criteria. *Journal of Personality and Social Psychology*, 93(3), 431-446.
- Smith, M., Shneiderman, B., Milic-Frayling, N., Mendes Rodrigues, E., Barash, V., Dunne, C. (2009, June 25-27, 2009). *Analyzing (social media) networks with NodeXL*. presented at the meeting of the Proceedings of the fourth international conference on Communities and technologies, University Park, PA, USA. doi:10.1145/1556460.1556497.
- Solomon, R. C. (2007). *True to Our Feelings: What Our Emotions Are Really Telling Us*. Oxford, UK: Oxford University Press.
- Sowe, S., Stamelos, I., & Angelis, L. (2008). Understanding knowledge sharing activities in free/open source software projects: An empirical study. *Journal of Systems and Software*, 81(3), 431-446.
- Spence, D. P., Scarborough, H. S., & Hoff Ginsberg, E. (1978). Lexical correlates of cervical cancer. *Social Science & Medicine. Part A: Medical Psychology & Medical Sociology*, 12(0), 141-145. doi:10.1016/0271-7123(78)90042-1.
- Spinellis, D. (2006, May 23, 2006). *Global software development in the freeBSD project*. presented at the meeting of the Proceedings of the 2006 international workshop on Global software development for the practitioner, Shanghai, China. doi:10.1145/1138506.1138524.
- Stake, R. E. (1995). *The Art of Case Study Research*. Thousand Oaks, CA: Sage Publication.
- Standish Group. (1995). *The Chaos Report*. Retrieved May 1, 2006, from http://www.standishgroup.com/sample_research/PDFpages/chaos1994.pdf
- Standish Group. (2001). *Extreme Chaos*. Retrieved May 1, 2006, from http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf
- Standish Group. (2009). *CHAOS Summary 2009*. West Yarmouth, MA: The Standish Group International Inc.
- Stevens, T. K., & Henry, S. M. (1997). *Using Belbin's Role to Improve Team Effectiveness*: Virginia Polytechnic Institute & State University Blacksburg, VA, USA.
- Stone, L. D., & Pennebaker, J. W. (2002). Trauma in Real Time: Talking and Avoiding Online Conversations about the Death of Princess Diana. *Basic Appl. Soc. Psychol.*, 24, 172-182.
- Stone, L. D., & Pennebaker, J. W. (2002). Trauma in Real Time: Talking and Avoiding Online Conversations About the Death of Princess Diana. *Basic and Applied Social Psychology*, 24(3), 173-183. doi:10.1207/s15324834basp2403_1.

- SWEBOK. (2004). *Guide to the Software Engineering Body of Knowledge*. Retrieved from <http://www.computer.org/portal/web/swebok>
- Szulanski, G. (2000). The Process of Knowledge Transfer: A Diachronic Analysis of Stickiness. *Organizational Behavior and Human Decision Processes*, 82(1), 9-27. doi:10.1006/obhd.2000.2884.
- Tan, P.-N., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining*. Boston, USA: Addison-Wesley.
- Tashakkori, A., & Teddlie, C. (1998). *Mixed methodology: Combining qualitative and quantitative approaches*. Thousand Oaks, CA: Sage.
- Taylor, M. A., Reed, R., & Berenbaum, S. (1994). Patterns of Speech Disorders in Schizophrenia and Mania. *The Journal of Nervous and Mental Disease*, 182(6), 319-326.
- Thomas, M., & Hynes, C. (2007). The darker side of groups. *Journal of Nursing Management*, 15(4), 375-385. doi:10.1111/j.1365-2834.2007.00697.x.
- Tiwana, A. (2004). Beyond the black box: knowledge overlaps in software outsourcing. *IEEE Software*, 21(5), 51-58. doi:10.1109/ms.2004.1331302.
- Tjosvold, D. (2008). The conflict-positive organization: it depends upon us. *Journal of Organizational Behavior*, 29(1), 19-28. doi:10.1002/job.473.
- Trapnell, P. D., & Wiggins, J. S. (1990). Extension of the Interpersonal Adjective Scales to Include the Big Five Dimensions of Personality. *Journal of Personality and Social Psychology*, 59(4), 781-790.
- Treude, C. (2010, May 2-8, 2010). *The role of emergent knowledge structures in collaborative software development*. presented at the meeting of the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, Cape Town, South Africa. doi:10.1145/1810295.1810400.
- Trigo, A., Varajao, J., Soto-Acosta, P., Barroso, J., Molina-Castillo, F. J., & Gonzalvez-Gallego, N. (2010). IT professionals: an Iberian snapshot. *International Journal of Human Capital and Information Technology Professionals (IJHCITP)* 1(1), 61-75.
- Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological Bulletin*, 63(6), 384 - 399.
- Tukey, J. W. (1977). *Exploratory data analysis*. Reading, MA: Addison-Wesley.
- Uzzi, B., & Spiro, J. (2005). Collaboration and creativity: The small world problem. *AJS*, 111, 447-504.
- van den Hoof, B., de Ridder, J., & Aukema, E. (2004). Exploring the Eagerness to Share Knowledge: The Role of Social Capital and ICT in Knowledge Sharing. In M. H. Huysman & V. Wulf (Eds.), *Social Capital and Information Technology* (pp. 163- 186). Cambridge, MA: MIT Press.
- van den Hooff, B., & de Ridder, J. (2004). Knowledge sharing in context: the influence of organizational commitment, communication climate and CMC use on knowledge sharing. *Journal of Knowledge Management*, 8(6), 117-130.
- van den Hooff, B., & Hendrix, L. (2004, April 2-3, 2004). *Eagerness and willingness to share: the relevance of different attitudes towards knowledge sharing*. presented at the meeting of the Fifth European Conference on Organizational Knowledge, Learning and Capabilities, Innsbruck, Austria.

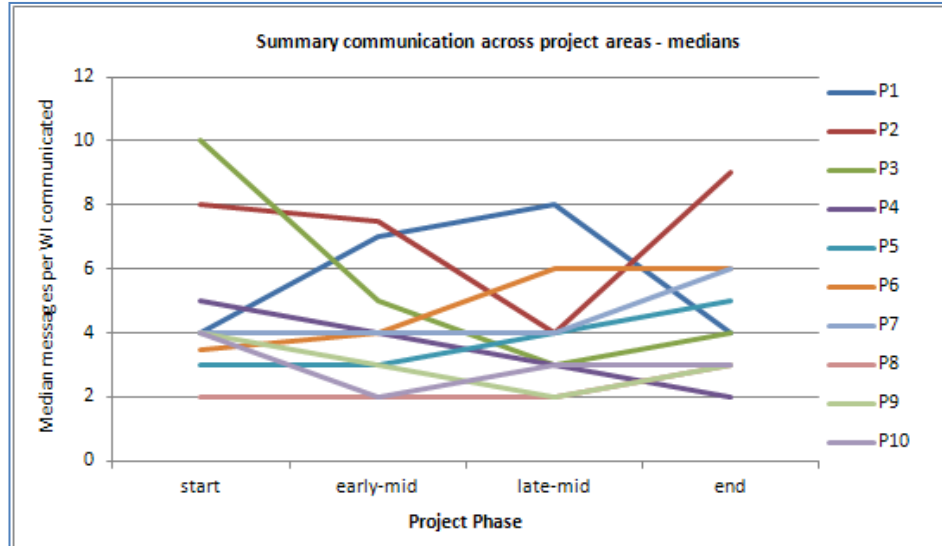
- Verner, J. M., & Evanco, W. M. (2005). In-house software development: what project management practices lead to success? *IEEE Software*, 22(1), 86 - 93
doi:10.1109/MS.2005.12.
- Vessey, I., Ramesh, V., & Glass, R. L. (2002). Research in Information Systems: An Empirical Study of Diversity in the Discipline and Its Journals. *Journal of Management Information Systems*, 19(2), 129-174.
- Vogt, W. P. (2005). *Dictionary of Statistics & Methodology: A Nontechnical Guide for the Social Sciences* (3rd ed.). Thousand Oaks: SAGE Publications, Inc.
- Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes* (14th ed.). Cambridge: Harvard University Press.
- Walle, T., & Hannay, J. E. (2009, October 15-16, 2009). *Personality and the nature of collaboration in pair programming*. presented at the meeting of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, Florida, USA.
doi:10.1109/ESEM.2009.5315996.
- Walsh, J. P. (1995). Managerial and Organizational Cognition: Notes from a Trip Down Memory Lane. *Organisation Science*, 6(3), 280-321. doi:10.2307/2635252.
- Walsham, G. (1993). *Interpreting Information Systems in Organizations*. New York: John Wiley & Sons, Inc.
- Walther, J., & Burgoon, J. K. (1992). Relational Communication in Computer-Mediated Interaction. *Human Communication Research*, 19(1), 50 -88.
- Walz, D., Elam, J., & Curtis, B. (1993). Inside a software design team: knowledge acquisition, sharing, and integration. *Commun. ACM*, 36(10), 63-77.
doi:10.1145/163430.163447.
- Wasserman, S., & Faust, K. (1997). *Social network analysis: methods and applications*. Cambridge: Cambridge University Press.
- Watson, W., & Michaelsen, L. (1988). Group Interaction Behaviors that Affect Group Performance on an Intellective Task. *Group & Organization Management*, 13(4), 495-516. doi:10.1177/105960118801300406.
- Weber, R. (1990). *Basic content analysis*. Beverly Hills, CA: Sage.
- Wieringa, R., Daneva, M., & Condori-Fernández, N. (2011, September 22-23, 2011). *The Structure of Design Theories, and an Analysis of their Use in Software Engineering Experiments*. presented at the meeting of the International Symposium on Empirical Software Engineering and Measurement (ESEM 2011), Banff, Canada. doi:10.1109/ESEM.2011.38.
- Willging, P. (2005). Using Social Network Analysis Techniques to Examine Online Interactions. *US-China Education Review*, 2(9), 46-56.
- Williams, L., & Kessler, R. (2003). *Pair Programming Illuminated*. Boston: Addison-Wesley.
- Wilson, M. (1987). MRC Psycholinguistic Database: Machine Usable Dictionary, Version 2.00.
- Wilson, M. (1988). The MRC Psycholinguistic Database: Machine Readable Dictionary, Version 2. *Behavioural Research Methods, Instruments and Computers*, 29(1), 6-11.

- Wohlin, C. (2002, August 23, 2002). *Is prior knowledge of a programming language important for software quality?* presented at the meeting of the Proceedings of the International Symposium of Empirical Software Engineering, 2002, Nara, Japan. doi:10.1109/isese.2002.1166922.
- Wolf, T., Schroter, A., Damian, D., & Nguyen, T. (2009, May 16-24, 2009). *Predicting build failures using social network analysis on developer communication.* presented at the meeting of the Proceedings of the 31st International Conference on Software Engineering, Vancouver, BC. doi:10.1109/icse.2009.5070503.
- Wolf, T., Schroter, A., Damian, D., Panjer, L. D., & Nguyen, T. H. D. (2009). Mining Task-Based Social Networks to Explore Collaboration in Software Teams. *IEEE Software*, 26(1), 58-66. doi:10.1109/MS.2009.16.
- Woodcock, M. (1989). *Team Development Manual*. Surrey, UK: Gower, Aldershot.
- Wynekoop, J., & Walz, D. (2000). Investigating traits of top performing software developers. *Information Technology & People*, 13(3), 186 - 195. doi:10.1108/09593840010377626.
- Yee, N., Harris, H., Jabon, M., & Bailenson, J. N. (2010). The Expression of Personality in Virtual Worlds. *Social Psychological and Personality Science*, 1(4). doi:10.1177/1948550610379056.
- Yen, M. Y. M., & Scamell, R. W. (1993). A human factors experimental comparison of SQL and QBE. *IEEE Transactions on Software Engineering*, 19(4), 390-409. doi:10.1109/32.223806.
- Yin, R. (2003). *Case Study Research: Design and Methods* (Third ed., Vol. 5). Thousand Oaks, CA: Sage Publications, Inc.
- Young, C., & Terashima, H. (2008, August 4-8, 2008). *How Did We Adapt Agile Processes to Our Distributed Development?* presented at the meeting of the Agile, 2008 Conference, Toronto, ON. doi:10.1109/Agile.2008.7.
- Yu, L., Ramaswamy, S., Mishra, A., & Mishra, D. (2011, October 17-21, 2011). *Communications in global software development: an empirical study using GTK+ OSS repository.* presented at the meeting of the Proceedings of the 2011th Confederated international conference on On the move to meaningful internet systems, Crete, Greece. doi:10.1007/978-3-642-25126-9_32.
- Zakaria, N., Amelinckx, A., & Wilemon, D. (2004). Working Together Apart? Building a Knowledge-Sharing Culture for Global Virtual Teams. *Creativity and Innovation Management*, 13(1), 15-29. doi:10.1111/j.1467-8691.2004.00290.x.
- Zalesny, M. D. (1990). Rater Confidence and Social Influence in Performance Appraisals. *Journal of Applied Psychology*, 75(3), 274-289.
- Zannier, C., Chiasson, M., & Maurer, F. (2007). A model of design decision making based on empirical results of interviews with software designers. *Information and Software Technology*, 49(6), 637-653. doi:10.1016/j.infsof.2007.02.010.
- Zeldow, P. B., & McAdams, D. P. (1993). On the Comparison of TAT and Free Speech Techniques in Personality Assessment. *Journal of Personality Assessment*, 60(1), 181-185.
- Zheng, J., Veinott, E., Bos, N., Olson, J. S., & Olson, G. M. (2002, April 20-25, 2002). *Trust without touch: jumpstarting long-distance trust with initial social activities.* presented at the meeting of the Proceedings of the SIGCHI conference

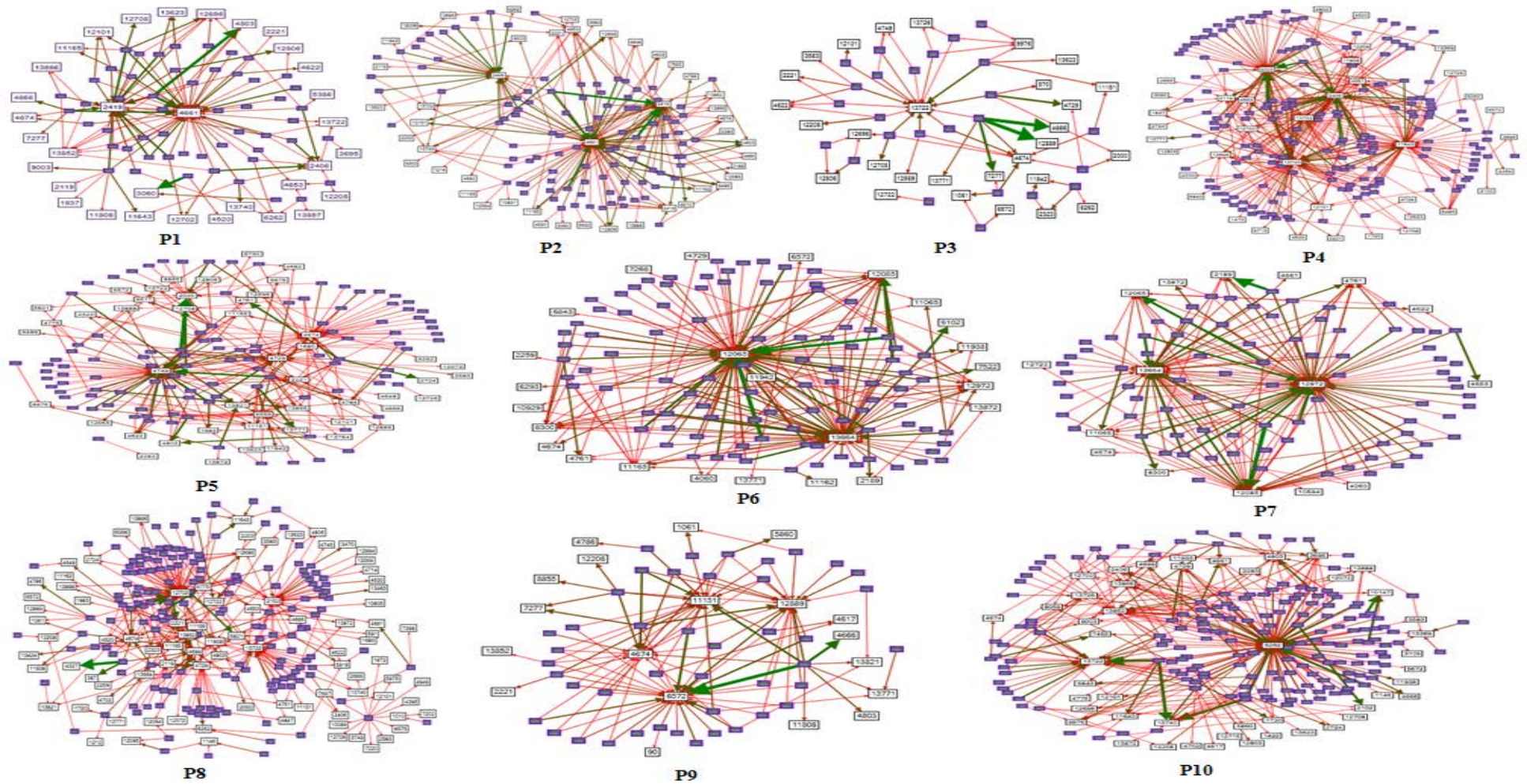
- on Human factors in computing systems: Changing our world, changing ourselves, Minneapolis, Minnesota, USA. doi:10.1145/503376.503402.
- Zhong, X., Huang, Q., Davison, R. M., Yang, X., & Chen, H. (2012). Empowering teams through social network ties. *International Journal of Information Management*, 32(3), 209-220. doi:10.1016/j.ijinfomgt.2011.11.001.
- Zhou, L., Burgoon, J. K., Zhang, D., & Nunamaker, J. F. (2004). Language dominance in interpersonal deception in computer-mediated communication. *Computers in Human Behavior*, 20(3), 381-402.
- Zhou, M., & Mockus, A. (2011, May 21-28, 2011). *Does the initial environment impact the future of developers?* presented at the meeting of the Proceedings of the 33rd International Conference on Software Engineering, Waikiki, Honolulu, HI, USA. doi:10.1145/1985793.1985831.
- Zhu, E. (1996). Meaning Negotiation, Knowledge Construction, and Mentoring in a Distance Learning Course. *Education Research Information Center*. Symposium conducted at the meeting of the Selected Research and Development Presentations at the 1996 National Convention of the Association for Educational Communications and Technology, Indianapolis, USA.
- Zigurs, I. (2003). Leadership in Virtual Teams: Oxymoron or Opportunity? *Organizational Dynamics*, 31(4), 339-351. doi:10.1016/S0090-2616(02)00132-8.
- Zimmermann, T., & Nagappan, N. (2008, May 10-18, 2008). *Predicting defects using network analysis on dependency graphs*. presented at the meeting of the Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany. doi:10.1145/1368088.1368161.
- Zimmermann, T., & Nagappan, N. (2009, October 15-16, 2009). *Predicting defects with program dependencies*. presented at the meeting of the 3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009., Lake Buena Vista, FL. doi:10.1109/ESEM.2009.5316024.
- Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schroter, A., & Weiss, C. (2010). What Makes a Good Bug Report? *IEEE Transactions on Software Engineering*, 36(5), 618-643. doi:10.1109/tse.2010.63.
- Zwikaël, O., & Ahn, M. (2010). The Effectiveness of Risk Management: An Analysis of Project Risk Planning Across Industries and Countries. *Risk analysis : an official publication of the Society for Risk Analysis, Wiley Online Library*, 31(1), 1 -12. doi:10.1111/j.1539-6924.2010.01470.x.

Appendices

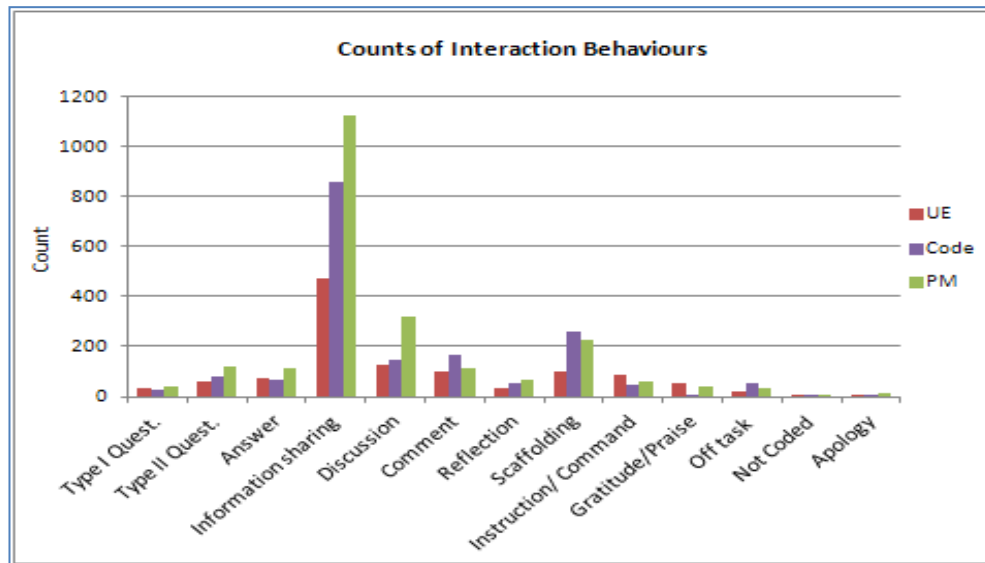
Appendix I. Median message per WI communicated over project phases (P1- P10)



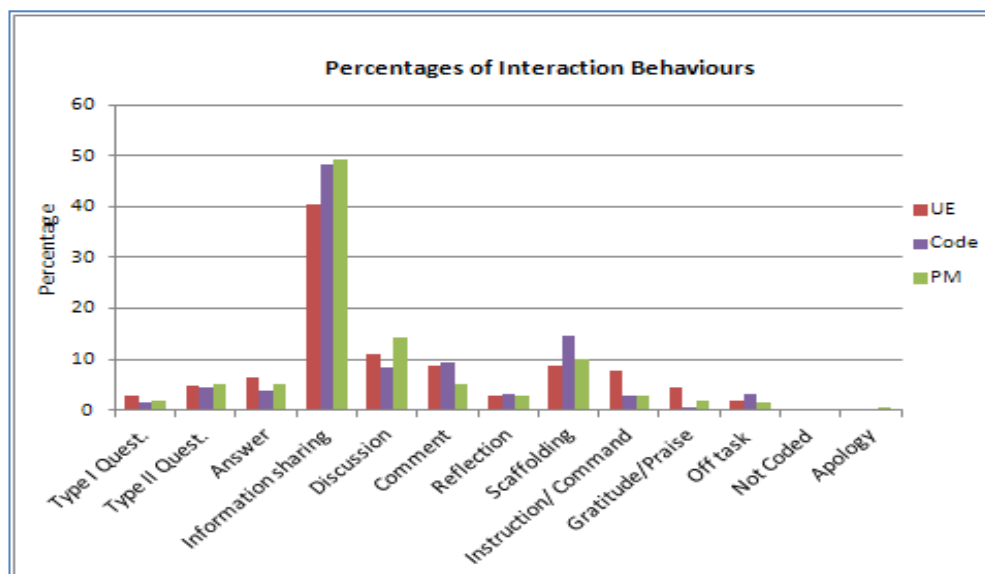
Appendix II. Sociograms for of all ten Jazz teams (P1 – P10)



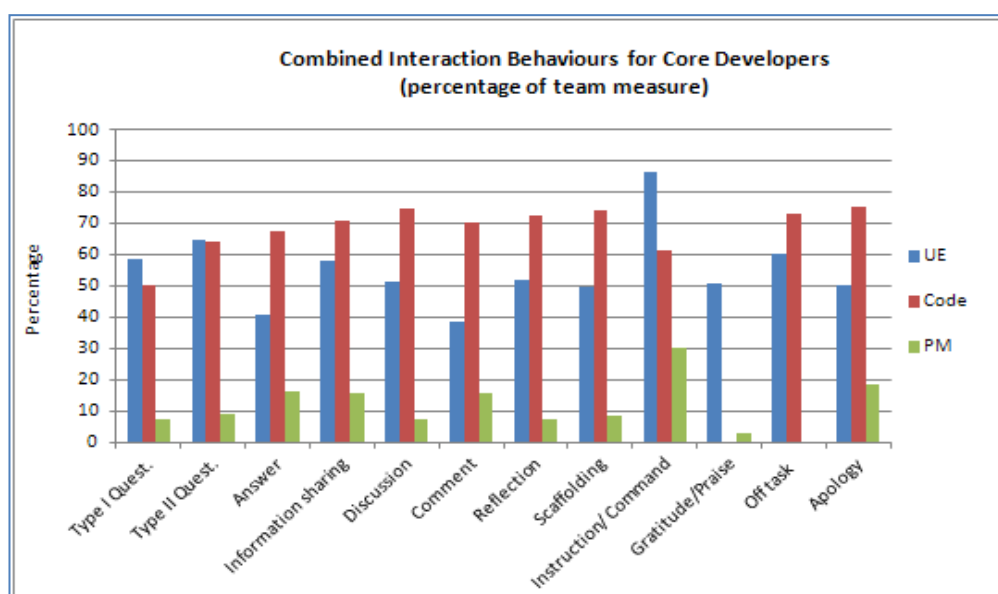
Appendix III. Interaction behaviours (counts) for the UE, Code and PM project practitioners



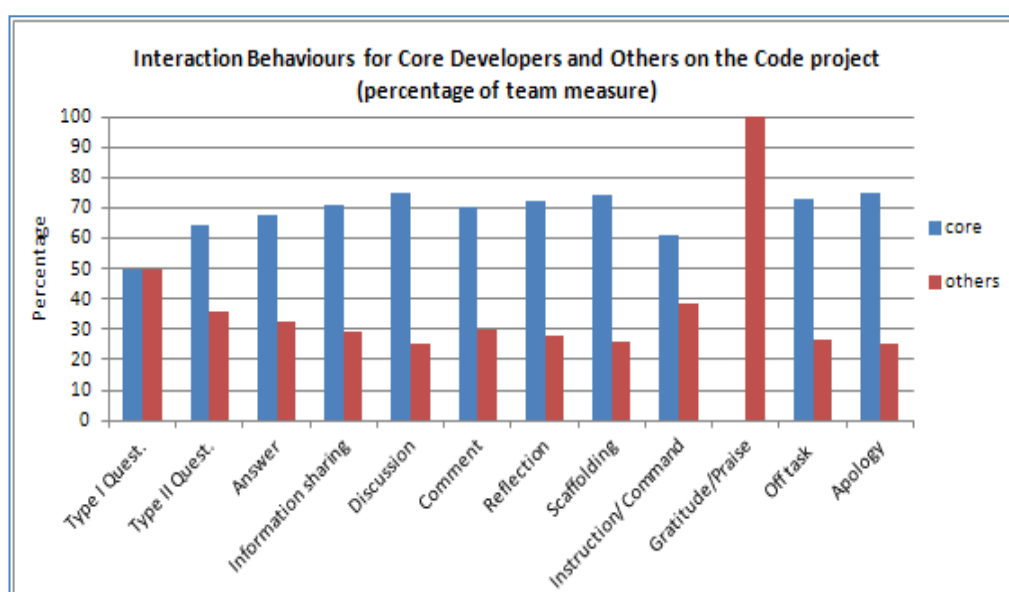
Appendix IV. Percentages of interaction behaviours across the UE, Code and PM project areas



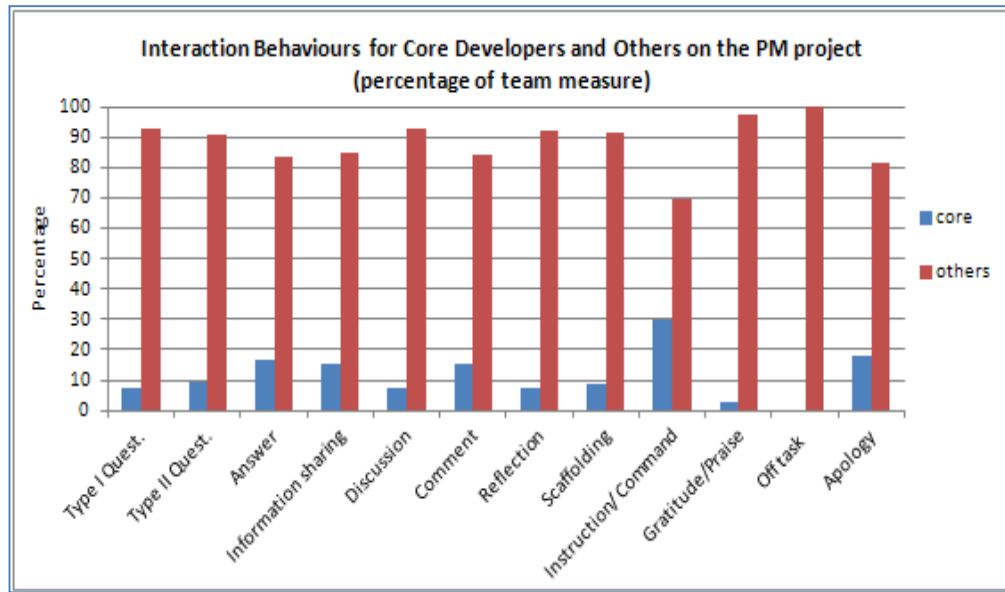
Appendix V. Combined percentages of overall project interaction behaviours for core developers



Appendix VI. Summary of project interaction for the core developers and others (for Code project area (P7))



Appendix VII. Summary of project interaction for the core developers and others (for PM project area (P8))

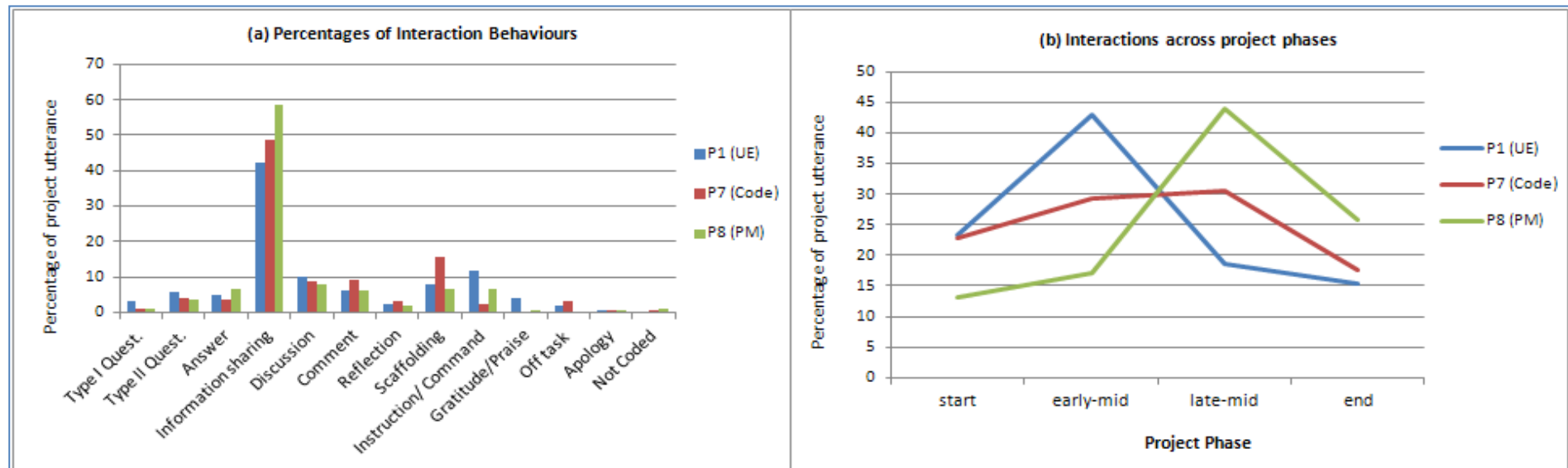


Appendix VIII. Descriptive statistics for core developers messages across the project phases

Phase	Messages (N)	Mean	Median	SD	SK	KS	Std. Error of SK	Std. Error of KS
start	657	65.7	63.0	37.2	0.71	-0.90	0.69	1.33
early-mid	703	70.3	72.5	50.4	1.18	2.03	0.69	1.33
late-mid	631	63.1	67.5	31.1	-0.59	-0.19	0.69	1.33
end	574	57.4	61.0	34.1	0.37	0.40	0.69	1.33
Mean	641.3	64.1	66.0	38.2	0.42	0.34	0.69	1.33

Notes: SD = Standard Deviation, SK = Skewness, KS = Kurtosis

Appendix IX. Aggregated interactions for core developers



Appendix X. Confidentiality Agreement



Agreement to Use Jazz Repository for Research

This is an agreement ("Agreement") entered into between **Sherlock Licorish (AUT)** ("Recipient") and International Business Machines Corporation ("IBM") through IBM Research, concerning the use of the Jazz Repository for research purposes and the appropriate measures of confidentiality associated with that use.

1. Licensee in Good Standing

Recipient must be a licensee in good standing under the License for Early Release of Jazz Programs and must maintain that status for the duration of this Agreement.

2. Repository and Analysis

IBM may make available a copy of the IBM Jazz project development repository ("Repository") under the terms of this Agreement. Repository may include software, information and other material including, without limitation, ideas, concepts, know-how and techniques. Repository may also include information about individuals including, without limitation, e-mail addresses, postal addresses and names of employers (collectively "Personal Information"). All information or materials, in all forms, derived, generated and/or collected from Repository, including, without limitation, methods, techniques, and/or programs and further including, without limitation, statistics, summaries, reports, studies, surveys, and visualizations shall constitute "Analysis".

3. Confidentiality

Repository and all Analysis shall be considered "Confidential Information". Recipient shall not disclose, publish, transmit, redistribute or otherwise make available Confidential Information.

Notwithstanding the foregoing, no confidentiality obligation will apply if Confidential Information is:

- i) already rightfully in the Recipient's possession or rightfully received by the Recipient without a nondisclosure obligation;
- ii) publicly available when Repository is received, or thereafter becomes publicly available through no fault of the Recipient;
- iii) disclosed to a third party by IBM without a nondisclosure obligation;
- iv) a publication made available pursuant to Section 4.

Aggregation in statistical form of

- i) project development behavior derived from Repository, of the nature of the number of comments in work items, and the number of work items of a particular status filed against a particular category, and/or
- ii) analysis of Repository artifacts, of the nature of the number of work items per change set, amount of code changed, sizes of files, number of methods per class, size or frequency of comments in code, percentage or size of code cloned, and number of clones per file, and under no circumstances containing any information pertaining to any code in any form in Repository

will not be considered Confidential Information, provided that such an aggregation contains no excerpts from the Repository, including information about individuals and/or their locations.

4. Publication of Analysis

The Recipient may include Analysis in an article for publication in scientific journals, periodicals, dissertations, conference papers, theses, reports or other research-oriented publications, subject to prior review and confirmation of compliance with the terms of this Agreement by IBM no less than thirty days before its submission for publication. Such articles may not contain Repository or any excerpts of Repository.

5. Use of Repository

Recipient may not:

- i) reverse assemble, reverse compile, or otherwise translate Repository into human-readable form or into another program language (except as may be specifically permitted by law without the possibility of contractual waiver);
- ii) use Repository or any information related to Repository to develop software or services that provide the same or similar functionality to the Program; or
- iii) use Repository on a service-bureau basis.

6. Use of Personal Information

Recipient may not use Personal Information for any purpose, including, without limitation, to identify or contact any individual or organization.

7. Term and Termination

Unless terminated earlier, this Agreement will begin on the date of execution and terminate one year thereafter. Recipient may request renewal of the Agreement for an additional term of equivalent duration, subject to IBM approval. Either party may terminate this Agreement upon thirty (30) days' written notice to the other party. Upon termination of the Agreement, Recipient must destroy Repository and any copies, as well as any Analysis in a tangible medium not included in a publication approved by IBM as described in section 4.0.

8. Survival

Sections 3.0, 4.0 and 6.0 shall survive termination of this Agreement.

9. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, REPOSITORY IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Recipient assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

10. Disclaimer of Liabilities

IBM SHALL HAVE NO LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF REPOSITORY OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

11. General Provisions

This Agreement does not confer any right to use in advertising, publications or promotional activities any name, trade name, trademark or other designation of either party

In the event that any provision of this Agreement is held to be invalid or unenforceable, the remaining provisions of this Agreement remain in full force and effect.

Recipient may not export Repository or take any action with respect to Repository that violates applicable export control laws.

Recipient may not assign this Agreement, in whole or in part, without IBM's prior written consent. Any attempt to do so is void.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America, without regard to conflict of law principles. The United Nations Convention on Contracts for the International Sale of Goods does not apply.

Recipient acknowledges having read the terms of the Agreement. By signing below, each party agrees to comply with the terms of the Agreement (and any subsequent amendment thereto).

ACCEPTED AND AGREED TO:		ACCEPTED AND AGREED TO:	
International Business Machines Corporation			
By:		By:	
_____	—	_____	—
IBM Signature		Signature	
_____	—	_____	—
Printed Name		Printed Name	
_____	—	_____	—
Title & Organiz		Title & Organizatic	
_____	—	_____	—
IBM Address:		Address:	