The Theory of One Team: Agile Software Development with Distributed Teams

by

Siva Dorairaj

A thesis submitted to the Victoria University of Wellington in fulfilment of the requirements for the degree of Doctor of Philosophy in Software Engineering.

Victoria University of Wellington 2013
Abstract

Team co-location is a hallmark of Agile software development that advocates face-to-face interaction and close collaboration among team members. Distributed teams, however, use Agile methods despite the separation of team members through space, time and culture. Little is known about how distributed teams use Agile methods for software development. A Grounded Theory research study that involved 55 participants from 38 different software companies in the USA, India, and Australia was carried out to investigate the key concern of distributed teams in Agile software development. This thesis proposes “The Theory of One Team” which explains how a distributed team in Agile software development adopts explicit strategies for bridging spatial, temporal, and socio-cultural distances, while facing critical impact factors, in order to become one team. This thesis primarily describes how a distributed team resolves the key concern of becoming one team. This thesis also provides the members of a distributed team with techniques for building trust with one another. In addition, this thesis serves to inform senior managers about the importance of supporting distributed teams in Agile software development.
Acknowledgments

“Interdependence is and ought to be as much the ideal of man as self-sufficiency. Man is a social being. Without interrelation with society he cannot realise his oneness with the universe or suppress his egotism. His social interdependence enables him to test his faith and to prove himself on the touchstone of reality.” —Mahatma Gandhi, Young India, March 21, 1929, p. 93.

Pranams to the Divine Mother, Satguru Mata Amritanandamayi Devi, The Hugging Saint, for Her grace. Amma, this PhD thesis would not have been possible without Your blessings.

In a research study that takes as long as this, there are always people to thank. In the past four years, there has always been someone prepared to help me. My apologies to those I may miss – a collective “Thank You” to all.

I would like to thank my parents, A.Dorairaj and S.Tanaletchemi, for their blessings and well-wishes; my wife and best friend, Rajeswary Nadarajan, for her love, sacrifices, and stoicism – thank you, Rajes, for carrying me through all this time; my children, Priya Darshini, Kaveesha Sree, and Ajay Sri Vishnu, for their affection and understanding.
I am deeply grateful to Professor James Noble for his excellent supervision, clear guidance, careful insight, and financial support in carrying out the research study and subsequently publishing research papers and completing this thesis. I am thankful to Dr. Petra Malik for her guidance and support. I am also thankful to Professor John Hine, Professor Dale Carnegie, and Dr. Peter Andreae (Pondy) for their support in many ways.

I wish to thank mentors and fellow researchers at VUW, which include but not limited to Dr. George Allan, Dr. Rashina Hoda, Dr. Diane Strode, Michael Waterman and Craig Anslow, for their support and encouragement. Sharing an office with Craig Anslow is an experience not to be forgotten! I would like to thank Dr. Monique Damitio, Dr. Rohitash Chandra, Suzan Hall, Mani Nambayah, Gina Straker, Michele de la Cour, and Prema Ram for their support in so many ways throughout this lengthy journey to pursue a PhD. To all my friends, both in and out of university, thank you for sharing the sorrow and happiness in my life. To the other postgraduates in the department, thanks for asking me to lunch!

I am grateful to Universiti Tenaga Nasional (Malaysia) for a PhD scholarship, and the Agile Alliance (USA), the J.L Stewart Scholarship and the Victoria University of Wellington for travel grants.

I thank my research participants for coming forth and so generously sharing their time and experiences with me – I owe a debt of gratitude that can never be repaid. I sincerely hope that the findings from this research will benefit the participants and the wider Agile community.

Last but not least, I thank Professor Philippe Kruchten (British Columbia University, Canada), Associate Professor David Parsons (Massey University, New Zealand) and Professor Sid Huff (Victoria University of Wellington, New Zealand) for their time and effort in examining this thesis.
## Contents

1 Introduction 1
   1.1 Research Contribution ............................. 3
   1.2 Structure of Thesis ................................. 4

2 Literature Review 7
   2.1 Agile Software Development ......................... 7
      2.1.1 Agile Values .................................. 8
      2.1.2 Agile Principles ................................ 9
      2.1.3 Agile Methods .................................. 12
      2.1.4 Agile Teams ................................... 22
   2.2 Distributed Teams .................................. 28
      2.2.1 Varieties of Team ................................ 28
      2.2.2 Global Distance ................................ 30
      2.2.3 Why Go Distributed? ............................ 31
   2.3 Concerns of Distributed Teams ....................... 33
      2.3.1 Promoting Communication ......................... 33
      2.3.2 Building Trust .................................. 34
      2.3.3 Managing Knowledge .............................. 36
      2.3.4 Bridging Cultural Differences ................... 38
   2.4 Discussion ......................................... 41

3 Research Design 47
   3.1 Research Paradigms .................................. 47
3.1.1 Positivist ................................................. 48
3.1.2 Interpretivist ............................................ 49
3.1.3 Postpositivist ........................................... 50
3.1.4 Critical Theory ........................................... 50
3.1.5 Selecting a research paradigm ......................... 51
3.2 Research Approaches ..................................... 52
3.3 Research Methods ........................................... 53
3.3.1 Case Study .............................................. 54
3.3.2 Ethnography ............................................. 54
3.3.3 Grounded Theory ........................................ 54
3.3.4 Selecting a research method ........................... 55
3.4 Role of the Researcher ..................................... 57
3.5 Grounded Theory ............................................ 59
3.5.1 Brief History ............................................ 59
3.5.2 Theoretical Sensitivity ................................ 61
3.5.3 Research Area .......................................... 62
3.5.4 Use of the Literature .................................. 63
3.6 Data Collection ............................................. 65
3.6.1 Theoretical Sampling .................................. 65
3.6.2 Recruiting Participants ................................. 68
3.6.3 Interviews and Observations ............................ 70
3.6.4 Research Context ..................................... 74
3.7 Data Analysis .............................................. 77
3.7.1 Open Coding .......................................... 79
3.7.2 Core Category ......................................... 82
3.7.3 Selective Coding ....................................... 84
3.7.4 Memoing ............................................... 84
3.7.5 Sorting ................................................ 88
3.7.6 Theoretical Coding .................................... 90
3.8 Generating a Theory ...................................... 92
3.9 Evaluating the Theory ..................................... 97
4 The Theory of One Team
4.1 Same Team
4.1.1 Knowing the Team
4.1.2 Engaging with the Team
4.1.3 Establishing Shared Vision
4.1.4 Team Recognition
4.2 Discussion

5 Bridging Temporal Distance
5.1 Same Time
5.1.1 Planned Formal Meetings
5.1.2 Planned Unconference
5.1.3 Spontaneous Conversations
5.1.4 Shifting Working Hours
5.1.5 Real-time Collaboration
5.2 Discussion

6 Bridging Spatial Distance
6.1 Same Space
6.1.1 Inception Workshops
6.1.2 Cross-Site Visits
6.1.3 Rotation
6.1.4 Coach Travels
6.1.5 Temporary Co-location
6.2 Discussion

7 Bridging Socio-Cultural Distance
7.1 Same Culture
7.1.1 Cultural Awareness
7.1.2 Language Support
CONTENTS

7.1.3 Open Communication ..................................... 174
7.1.4 Shared Value .............................................. 178

7.2 Same Practices ...................................................... 180
7.2.1 Shared Standards ........................................... 181
7.2.2 Shared Knowledge ......................................... 184
7.2.3 Shared e-Workspace ....................................... 187
7.2.4 Shared Pain .................................................. 188

7.3 Discussion ........................................................... 190

8 Impact Factors ....................................................... 193
8.1 Trust ................................................................. 193
8.1.1 Social Communication .................................... 194
8.1.2 Evidence of Expertise ...................................... 196
8.1.3 Collaboration Initiatives ................................... 197
8.1.4 Regular Communication .................................. 200
8.1.5 Substantive Feedback ..................................... 202
8.1.6 Team above Self ............................................ 203
8.1.7 Communication with Concern ............................ 204
8.2 Senior Management Support ................................. 206
8.2.1 Organisational Culture .................................. 206
8.2.2 Human Resource Management ......................... 208
8.2.3 Financial Sponsorship .................................... 209
8.2.4 Technical Infrastructure .................................. 211
8.2.5 Customer Liaison .......................................... 212

8.3 Discussion ........................................................... 213
8.3.1 Trust ........................................................... 214
8.3.2 Senior Management Support ............................. 217

9 Agility in Context ..................................................... 219
9.1 Time Overlap ...................................................... 220
9.2 Number of Sites .................................................. 222
9.3 Team Size .......................................................... 224
## CONTENTS

9.4 Technical Infrastructure ........................................ 226  
9.5 Project Criticality ............................................. 228  
9.6 Discussion ....................................................... 230  

10 Discussion ......................................................... 233  
10.1 Nature of contribution .......................................... 233  
10.2 Relationships between strategies ............................... 236  
10.2.1 Same Team strategy ......................................... 237  
10.2.2 Same Time strategy ......................................... 239  
10.2.3 Same Space strategy ......................................... 241  
10.2.4 Same Culture strategy ...................................... 243  
10.2.5 Same Practices strategy ................................. 244  
10.3 Applicability of the theory to co-located teams ............. 246  
10.4 Observational experience ..................................... 248  
10.5 Selecting a theoretical coding family ......................... 253  
10.5.1 System Parts ............................................... 254  
10.5.2 Six C .................................................... 255  
10.5.3 Strategy .................................................. 257  
10.6 Revisiting the Agile Manifesto ................................. 258  
10.7 Evaluating our theory ......................................... 261  
10.7.1 Glaser’s approach .......................................... 262  
10.7.2 Angen’s approach ......................................... 264  
10.7.3 Lincoln and Guba’s approach ............................ 265  
10.7.4 Weber’s approach ........................................ 268  

11 Conclusion .......................................................... 277  
11.1 Research Contribution ........................................ 277  
11.2 Limitations ....................................................... 280  
11.3 Future Work ..................................................... 282
List of Figures

2.1 A Typical Scrum Sprint [54] ................................. 14
2.2 Roles in a Scrum Team ........................................ 16
2.3 XP Roles ......................................................... 19
2.4 XP Practices ...................................................... 21
2.5 Development stages of a team (adapted from Tucker [301]) . 24
2.6 Team Performance Model (adapted from Drexler et al. [70]) . 25
2.7 Varieties of Team (adapted from Lipnack and Stamps [192]) 29

3.1 Selective and theoretical sampling in Grounded Theory (adapted from Urquhart, Lehmann and Myers [302]) .............. 67
3.2 Typical Team Distributions. ................................. 70
3.3 Scrum Artifact: A Sprint Backlog. ............................ 74
3.4 Scrum Artifact: A Burndown Chart. .......................... 76
3.5 Example of emergence of a concept from codes ................ 80
3.6 Example of emergence of a category from underlying concepts 81
3.7 Levels of Abstractions in Grounded Theory .................. 82
3.8 Concept Memo on Team Interaction .......................... 86
3.9 Concept Memo on Planned Unconference ..................... 87
3.10 Process Memo on Participant Position: Senior Manager ...... 87
3.11 Process Memo on Theoretical Coding: System Part ........... 88
3.12 Theoretical sorting of memos. ................................ 89
3.13 Emergence of the Theoretical Code “Strategies” ............. 91
3.14 Emergence of the Theoretical Code “Impact Factors” ....... 91
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.15</td>
<td>Examples of theoretical coding families.</td>
<td>93</td>
</tr>
<tr>
<td>3.16</td>
<td>The progression of theory development in Grounded Theory (adapted from [186]).</td>
<td>94</td>
</tr>
<tr>
<td>4.1</td>
<td>The Theory of One Team</td>
<td>108</td>
</tr>
<tr>
<td>4.2</td>
<td>Plans and Goals of the Strategies.</td>
<td>109</td>
</tr>
<tr>
<td>4.3</td>
<td>The Theory of One Team in detail.</td>
<td>111</td>
</tr>
<tr>
<td>4.4</td>
<td>Tactics for the Same Team strategy</td>
<td>113</td>
</tr>
<tr>
<td>5.1</td>
<td>Tactics for the Same Time strategy</td>
<td>129</td>
</tr>
<tr>
<td>6.1</td>
<td>Tactics for the Same Space strategy</td>
<td>148</td>
</tr>
<tr>
<td>7.1</td>
<td>Tactics for the Same Culture strategy</td>
<td>167</td>
</tr>
<tr>
<td>7.2</td>
<td>Tactics for the Same Practices strategy</td>
<td>181</td>
</tr>
<tr>
<td>10.1</td>
<td>An example of distribution without time overlap</td>
<td>246</td>
</tr>
<tr>
<td>10.2</td>
<td>Creating time overlap</td>
<td>247</td>
</tr>
<tr>
<td>10.3</td>
<td>Scrum artifacts.</td>
<td>250</td>
</tr>
<tr>
<td>10.4</td>
<td>Scrum task board.</td>
<td>251</td>
</tr>
<tr>
<td>10.5</td>
<td>CollabNet ScrumWorks Pro</td>
<td>252</td>
</tr>
<tr>
<td>10.6</td>
<td>Atlassian Jira</td>
<td>252</td>
</tr>
<tr>
<td>10.7</td>
<td>Thoughtworks Mingle</td>
<td>253</td>
</tr>
<tr>
<td>10.8</td>
<td>Theoretical coding family: System Parts</td>
<td>255</td>
</tr>
<tr>
<td>10.9</td>
<td>Theoretical coding family: Six C</td>
<td>256</td>
</tr>
<tr>
<td>10.10</td>
<td>Theoretical coding family: Strategy</td>
<td>258</td>
</tr>
<tr>
<td>10.11</td>
<td>Weber’s criteria through Grounded Theory lens</td>
<td>269</td>
</tr>
<tr>
<td>10.12</td>
<td>Definition of substantive and theoretical codes.</td>
<td>270</td>
</tr>
<tr>
<td>10.13</td>
<td>Definition of terms.</td>
<td>271</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Summary of Participants, Teams, and Projects. (Agile Role: Scrum Master (SM), Agile Coach (AC), Developer (DEV), Business Analyst (BA), Quality Analyst (QA), Product Owner (PO), Senior Manager (MGT)) ................................. 75

3.2 Examples of Key Points and Codes - Data from Participant P1 80
Chapter 1

Introduction

Team co-location is a hallmark of Agile software development [53, 129]. Co-location promotes face-to-face conversation, fosters close collaboration, and maximises knowledge sharing between members of Agile teams [53, 129]. A co-located Agile team strives to provide business value to customer on a regular basis [51, 221].

In recent years, software companies are increasingly venturing into distributed software development in order to capitalise on the global resources pool, acquire an appropriate mix of expertise, lower costs, and reduce time-to-market [180, 280]. Distributed teams adopt Agile methods despite the separation of team members through space, time and culture in order to acquire similar benefits as using Agile methods in software development with co-located Agile teams [153, 170]. Distributed teams, however, face challenges to understand how to combine Agile methods with distributed software development due to spatial, temporal, and socio-cultural distances [149, 153, 171, 280]. It is crucial to understand the overall combination of the two approaches in order to benefit from using Agile methods in distributed software development. Several studies [162, 230, 280] suggest that there is a critical need for an in-depth study to discover how distributed teams use Agile methods while separated through space, time and culture.
Šmite et al. [280] argue there is still no consensus or deep, theoretically grounded, understanding of the applicability of Agile methods to distributed software development and flexibility in application of Agile methods in order to realise the benefits expected from combining Agile methods with distributed software development. Šmite et al. [279] suggest future research in Agile software development needs to study the combination of Agile teams from different geographical, cultural and temporal areas in order to develop theoretical models of Agile and distributed software development.

Taylor et al. [297] conducted a study on the usefulness of published research on global software development for Agile teams. They argue the published research is of minimal value to practitioners and does not provide much guidance particularly for distributed teams in Agile software development. Jalali and Wohlin [162] found the majority of the existing literature on Agile software development with distributed teams is in the form of experience reports. These researchers suggest the future for Agile teams in distributed software development research needs to build upon rigorous and thorough empirical research. There is clearly a need to conceptualise and theorise the underpinnings of distributed teams in Agile software development.

Dybå and Dingsøyr [73] report that only 36 out of 1996 studies of Agile software development (of all types of projects, with vast majority co-located) were found to be “research studies of acceptable rigour, credibility, and relevance”. There is little empirical knowledge on Agile software development with distributed teams [162]. Hence, more empirical research on this area would benefit the Agile teams and software companies that are considering using Agile methods in distributed software development [162, 170, 230]. Based on a large-scale empirical research study, this thesis explains how distributed teams in Agile software development bridge spatial, temporal, and socio-cultural distances in order to become one team.
1.1 Research Contribution

This research is based on interviews with 55 participants and observations on seven Agile teams from 38 different software companies in the USA, India, and Australia. “The Theory of One Team” explains how a distributed team in Agile software development adopts explicit strategies for bridging spatial, temporal and socio-cultural distances, while facing critical impact factors, in order to become one team. The main contributions of this thesis are as follows:

- This thesis presents the Same Team, Same Time, Same Space, Same Culture, and Same Practices strategies which describe how a distributed team in Agile software development bridges spatial, temporal, and socio-cultural distances in order to work together as one team. In particular, the Same Team strategy describes how to promote cohesion among members of a distributed team; the Same Time strategy describes how to promote synchronous communication; the Same Space strategy describes how to promote temporary physical co-location; the Same Culture strategy describes how to promote cross-cultural communication; and the Same Practices strategy describes how to promote the use of common practices across the different sites of a distributed team.

- This thesis describes the importance of trust between members of a distributed team in order to work together as one team. In particular, this thesis presents techniques for building trust in Agile software development with distributed teams.

- This thesis describes the importance of senior management support for distributed teams in Agile software development in order to work together as one team.
• This thesis describes a contextual approach to adopting the Same Team, Same Time, Same Space, Same Culture, and Same Practices strategies for a distributed team.

This thesis also provides a detailed description of how we used the Grounded Theory research method in this software engineering research study.

1.2 Structure of Thesis

This thesis has the following chapters:

• **Literature Review:** In chapter 2, we describe Agile software development from the perspective of Agile Manifesto, Agile methods, and Agile teams. We also describe distributed software development and the spatial, temporal and socio-cultural distances.

• **Research Design:** In chapter 3, we present an overview of the research design which integrates the research paradigm, research approach and research method. We also present an overview of the participants and projects, describe the data collection and analysis processes, and discuss the evaluation of the emergent theory.

• **The Theory of One Team:** In chapter 4, we present an overview of the grounded theory and describe the Same Team strategy for promoting team cohesion among members of a distributed team.

• **Bridging Temporal Distance:** In chapter 5, we describe the Same Time strategy for promoting synchronous communication in order to bridge the temporal distance.

• **Bridging Spatial Distance:** In chapter 6, we describe the Same Space strategy for promoting temporary physical co-location in order to bridge the spatial distance.
1.2. **STRUCTURE OF THESIS**

- **Bridging Socio-cultural Distance**: In chapter 7, we describe both the Same Culture strategy for promoting cross-cultural communication and the Same Practices strategy for promoting the use of common practices across different locations of a distributed team in order to bridge the socio-cultural distance.

- **Impact Factors**: In chapter 8, we describe how trust and senior management support impacts Agile software development with distributed teams.

- **Agility in Context**: In chapter 9, we discuss a contextual approach to adopting the Same Team, Same Time, Same Space, Same Culture, and Same Practices strategies for a distributed team.

- **Discussion**: In chapter 10, we discuss several important aspects of the grounded theory and describe in detail how the grounded theory was validated and evaluated.

- **Conclusion**: In chapter 11, we highlight the main contributions, discuss the limitations of the study, and finally present suggestions for future work.

The term “we” used in this thesis refers to Siva Dorairaj, typically in consultation with his supervisors; the term “our” is used to differentiate this thesis from other research in the discussion sections 4.2, 5.2, 6.2, 7.3, 8.3, 9.6, and the chapters 10 and 11.

In chapters 4–9, we included selected quotations drawn from the interviews that shed particular light on the findings. The quotations are presented verbatim from the interview transcripts with square brackets used to insert missing words to fix grammar or to anonymise participant details (such as names of individuals or companies). Three full stops (...) indicate combining two sentences referring to the same context but derived from different parts of the same interview.
Throughout the thesis, cross-references have been included to tie the related findings with one another or the discussion with the findings. The cross-reference is provided in brackets with the letter ‘s’ followed by a section number. For example, (s 4.1) indicates section 4.1.
Chapter 2

Literature Review

In this chapter, we describe Agile software development from the perspective of the Agile values, principles, methods and teams. We then describe the distributed teams and the spatial, temporal and socio-cultural distances. Finally, we describe the key challenges faced by distributed teams in Agile software development in the light of existing literature.

2.1 Agile Software Development

Agile is an umbrella term for a software development philosophy that encompasses the values, principles, methods, and teams [2, 51, 130, 129, 266]. Agile software development defines “a strategic capability, a capability to create and respond to change, a capability to balance flexibility and structure, a capability to draw creativity and innovation out of a development team, and a capability to lead organizations through turbulence and uncertainty” [130]. In general, Agile software development focuses on “feedback and change” [312], “adapting to change and delivering products of high quality through simple work-processes” [67], and “the value competent people and their relationships bring to software development” [220]. A detailed description of the Agile values, Agile principles, Agile methods, and Agile teams follows.
2.1.1 Agile Values

In February 2001, seventeen independent software development practitioners gathered to recognise the core values of several “easy to follow” software development methods, and wrote the Agile Manifesto [24]. The Agile Manifesto [24] describes the Agile values and a set of 12 Agile principles for Agile teams. The Agile Manifesto reads, in its entirety, as follows:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and Interactions over Processes and Tools
- Working Software over Comprehensive Documentation
- Customer Collaboration over Contract Negotiation
- Responding to Change over Planning

That is, while there is value in the items on the right, we value the items on the left more.

The first value emphasises the importance of interaction between individuals in a team. Despite the fact that tools and processes are important, however, the most important factor to consider is how the individuals in a team work together in order to develop software for customer. The second value describes the primary goal of software development is to develop valuable software rather than documents in order to provide value to customer. Though documentation is a valuable guide for understanding why a system is built or how to work with the system, it is never as important as the working software itself. The third value emphasises the importance of close collaboration between the development team and customer. Though a contract, in general, is an important document which describes the rights and responsibilities of the development team and customer, the development team should not focus on negotiating for a favourable contract but
rather focus on how to work together with customer in order to deliver what is needed for the customer. The final value reminds the team that change is an inevitable reality of software development. Hence, a prompt response to change, even late in development, is still sensible in order to remain relevant to the customer.

2.1.2 Agile Principles

The 12 principles behind the Agile Manifesto [24] provide the software development teams with an in-depth understanding of what Agile software development is all about. Understanding these principles allows development teams to uphold the Agile values during software development. A detailed description of the Agile principles follows.

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software: Customers are given the paramount importance in Agile software development. While customers may not know what they want until they see working software [122, 172, 199], the development teams deliver incremental working features of the final software product to the customers on an ongoing basis. When the customers use the features early in the development lifecycle, they could give continual feedback to the development team and gain business value even before the final software product is completed.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage: The development teams understand that the purpose of software development is to provide business value to customer. Therefore, changes that may happen on project requirements are not considered as impediments for the development teams but rather as an opportunity to increase the customer’s competitive advantage [12, 96].
3. **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale:** On short and frequent intervals, the development team delivers the incremental working features of the final software product to customer in order to get early feedback on those features [12, 96].

4. **Business people and developers must work together daily throughout the project:** Customer or the subject matter expert from the business domain continually works together with the development team throughout the entire duration of a project in order to develop a working software which fulfills customer requirements. The level of collaboration between the customer and the development team often determines the success or failure of a project [138, 199].

5. **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done:** Members of a development team ideally should feel motivated to work with one another and to engage with customer on a regular basis. It is a fundamental responsibility of a manager to provide the development team with the infrastructure and technology needed for software development, including proper tools for communication and collaboration. The managers must trust that the development team will work together in the best interest of the customer and the organisation [204, 216].

6. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation:** Face-to-face conversation provides team members with opportunities to understand not only verbal communication but also non-verbal cues and behaviours such as voice tone, body language, and hand gestures [208, 240]. Through face-to-face conversation, team members convey information and share knowledge efficiently and effectively with one another [19, 281, 305].
7. Working software is the primary measure of progress: Central to customer satisfaction is the development of a working software. The primary evaluation of the progress of projects and the performance of development teams are based on the successful development and delivery of working software [12, 96].

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely: While working at a constant pace throughout the project, a development team is able to deliver valuable software on a regular basis. A balance between work and life is important to keep a team together while maintaining a consistent progress on a project [146, 183].

9. Continuous attention to technical excellence and good design enhances agility: With the focus on technical excellence, a development team continually strives to improve the development processes and practices used in a project. While developing software, team members avoid complex designs and abstract meta-models but rather focus on good designs based on coding and modeling guidelines [12, 96].

10. Simplicity—the art of maximizing the amount of work not done—is essential: One of the important requirements of Agile methods is to infuse simplicity in development. Using simple approaches in software development increase the changeability of designs and processes. Agile methods encourage development to restrict to what is absolutely required in order to provide value for customers. Minimising the unnecessary overhead helps in keeping designs simple [12, 96].
11. The best architectures, requirements, and designs emerge from self-organising teams: A self-organising team coordinates software development activities and strives for continuous improvement without much intervention from the managers [137]. When a team is empowered to take decisions and given the freedom to carry out the work, the team performs at its best when developing the software.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly: The development team reflects on the processes and practices in place within the team on an ongoing basis. While the team members understand the importance of providing honest feedback with regard to the overall performance of the team, they remain open and transparent with one another in order to recognise the strengths and weaknesses of the entire team [12, 96].

2.1.3 Agile Methods

Agile methods are a family of software development methods that follow an iterative and incremental style of software development [2, 191]. With a growing number of Agile methods, the core methods includes XP (eXtreme Programming) [22, 23], Scrum [266], ASD (Adaptive Software Development) [132], DSDM (Dynamic Systems Development Method) [287], FDD (Feature Driven Development) [50, 232], and Crystal Clear [52].

Agile methods advocate face-to-face communication and close collaboration practices [51]. While some Agile methods focus on project management and collaboration practices (ASD, Scrum, DSDM), others focus on software development practices (XP, FDD). Though practices within each individual method vary, Agile methods, in general, fulfill the following aspects [130]:

...
Visioning: Project vision is shared between development teams and the customers in order to provide clear goals for projects and to keep the development teams focused on delivering business value to the customers.

Project inception: A comprehensive project information which includes project scope, objectives, constraints, business values, costs, schedule, success measures, technology assumptions, and rules of engagement, is documented and shared within the team.

Short, iterative, incremental development: A development team focuses on developing small features and deliver them frequently, while continually integrating those features onto the large software, in order to give immediate business values to customers.

Constant feedback: There are practices in place to foster constant feedback in order to synchronise work and share information.

Customer collaboration: Regular interaction between customers and the development teams fosters close collaboration for the purpose of developing shared understanding and knowledge sharing.

Technical excellence: With the focus on developing quality software in order to develop competitive advantage for the customers, practices are in place to create and maintain technical excellence on the software product.

Scrum and XP are considered to be the most widely adopted Agile methods in software development projects [91, 240]. A detailed description of Scrum [266] and XP [22, 23] follows.
Scrum

Scrum is not an acronym but a term adapted from an action called a scrum-down in rugby which refers to “a tight formation of forwards who bind together in specific positions to get an out-of-play ball back into play” [188, p.41]. Scrum was first articulated and observed by Takeuchi and Nonaka [295], but later formalised by Sutherland and Schwaber [265].

Scrum mainly provides techniques for project management. Scrum splits projects into a series of consecutive short work cycles called *sprints*, which are typically 2 to 4 weeks in duration, in order to develop deliverables [266]. During each sprint, the development teams carry out tasks to complete a small set of features – everything from designing to coding to testing – in order to deliver the features to customers [249]. Figure 2.1 illustrates a typical Scrum sprint.

![Figure 2.1: A Typical Scrum Sprint](54)

While the primary artifact of a Scrum is the final product itself, there are three other important artifacts used in the sprints:
2.1. **AGILE SOFTWARE DEVELOPMENT**

- **Product Backlog** is the complete list of prioritised features – including requirements, enhancement requests, usability improvements, and bugs – that a customer has requested for development. The product owner product continually prioritises the product backlog so that the development team always works on the most valuable features first.

- **Sprint Backlog** is a subset of the product backlog. The sprint backlog contains the highest priority features which have been assigned to a sprint.

- **Burndown Chart** shows the tasks remaining within the sprint on a daily basis and guides the development teams to track sprint progress and to decide when items must be removed from the sprint backlog and deferred to the next sprint.

A Scrum Team has three core roles: **Product Owner, Scrum Master, and Development Team** [293]. These core roles includes individuals who are directly involved and committed to the project, while ancillary roles such as managers and stakeholders are not directly involved in the project but often provide significant support for the Scrum team. Figure 2.2 illustrates a roles in a Scrum team. A brief description of the roles in the Scrum team is provided below.

- **Product Owner** is a customer representative who conveys the customer’s vision to the team. The product owner is responsible for providing requirements to define the features and prioritising the features to facilitate the development of the software product. When the features have been completed, the product owner may accept, reject, or request for modification depending on whether the features provide value to customers.
Figure 2.2: Roles in a Scrum Team

- **Scrum Master** is responsible for facilitating the software development project and mentoring the development teams. A Scrum Master focuses on the development process while removing any impediments to progress, facilitating meetings, and providing support to the development team to achieve the highest level of performance.

- **Development Team** is a cross-functional, self-organising team which typically comprises seven members, plus or minus two. Although a development team comprises software engineers, architects, programmers, and testers, every member of the development team is called a developer [293]. The development team has autonomy to determine how and when to complete the tasks assigned for a sprint.

Scrum advocates frequent communication through its four process activities, known as Scrum ceremonies, to keep the software development activities on track [266]. The four Scrum ceremonies are:
2.1. AGILE SOFTWARE DEVELOPMENT

- **Sprint planning meeting** indicates the start of a sprint where a development team meets with the customer, or product owner, in order to choose a small set of features to deliver during a sprint. When the customer or product owner describes the highest priority features to the team members, communication between them turns the high-level requirements into a detailed tasks that will be carried out during that sprint. The development team estimates the effort for the tasks and designs a detailed plan for achieving the goals of the sprint.

- **Daily Scrum**, or daily stand-up meeting, is a short meeting, often scheduled for fifteen minutes, which provides avenue for the development team members and customer to meet on a daily basis in order to synchronise the tasks in a sprint. During the daily Scrum, team members answers three Scrum questions: *What did I do yesterday? What will I do today? What impediments are in my way?*

- **Sprint review** is a product demonstration meeting where a development team shows the product completed during the sprint to the customer, or product owner. During a sprint review, the project is assessed based on the features completed in that sprint against the sprint goals determined during the sprint planning meeting.

- **Sprint retrospective** is a reflection meeting where the team members explore ways to achieve continuous success in future sprints and the overall project. With a focus on the major practices and processes used in a sprint, the team members describe what was done well (and what was not) and suggest potential improvements which may be implemented in future sprints.

Communication during Scrum ceremonies help to ensure that the most valuable features have been completed when the software development project ends. The success of Scrum ceremonies, however, depends on the close collaboration between individuals in the Scrum Team [293].
eXtreme Programming (XP)

eXtreme programming (XP) was developed by Kent Beck, Ward Cunningham, and Ron Jeffries [130]. While many software development projects are technically sophisticated, traditional practices face difficulties to address the challenges in software development [221]. XP provides software development teams with values, roles, and practices which guides the team members to carry out software development activities effectively while keeping the project on target [28, 264]. A description of XP roles, values and practices based on literature [23] is provided below.

There are six core XP roles: Customer, Coach, Tracker, Programmer, Tester, and Big Boss [22, 200]. While the development team consists of coach, tracker, programmer, and tester, XP requires the customer to be part of the project team and provide instant feedback on the development of the features for a project [22]. Figure 2.3 illustrates the XP Development Team and XP Project Team.

- **Customer** is responsible for defining what is the right software product for an organisation to gain competitive advantage, to determine the order in which features should be developed, and to make sure the features actually work as requested.

- **Coach** is responsible for keeping team members focused on process while mentoring them to learn and improve on the practices and processes in place within the team. The coach should possess technical and soft skills in order to influence the actions of the team members.

- **Tracker** is responsible for tracking the release plan (user stories), the iteration plan and the acceptance tests. While tracking is crucial to monitor the project progress, the tracker should have the ability to collect these information without disturbing the process as a whole.
• *Programmer* is responsible for designing a solution and writing the code to develop the features to support the user stories.

• *Tester* is responsible for writing unit tests for every functionality and assist customers to define and write acceptance tests for user stories.

• *Big Boss*, often known as the project sponsor, is responsible for the overall project. The project team keeps frequent communication with the Big Boss in order to report the progress on the project and acquire continuous support such as financial support or organisational support [200].

A set of five values guides the adoption of suitable practices for the development teams [23]. XP values closely relate to the values and principles of Agile software development [24].
• **Communication**: XP advocates continuous communication between customers and development teams. Regular communication facilitate knowledge sharing, particularly technical knowledge within development teams, and business knowledge within the project teams.

• **Simplicity**: Simplicity focuses on developing simple working software earlier rather than complex software later, in order to provide immediate business values to customer. Designing solutions and coding should be based on simple architectures.

• **Feedback**: Through fast and substantive feedback from customers with regard to the features implemented in a project, the development teams strive for continuous improvement of the project.

• **Courage**: A team member should have the courage to improve the code written by another member of the team. All team members have rights to whole project code-base and to improve them as necessary.

• **Respect**: Customers, managers, and development team members should keep honest communication and trust one another while working together in a project. Mutual respect and trust between them fosters close collaboration while carrying out the project activities.

While XP practices evolved over the past several years, the rationale of introducing a practice is to achieve the XP values [22, 23]. Figure 2.4 illustrates the XP Practices.

• **Planning game** is a planning meeting for customers decide the scope, priorities, and timing of releases based on programmers’ estimation on the effort required for completing the tasks.

• **Small releases** encourage simple working features to be delivered on a frequent and regular basis, while integrating those features into the large software product.
2.1. AGILE SOFTWARE DEVELOPMENT

Figure 2.4: XP Practices

- **Metaphor** is an abstract story of the software product which is shared between the customer and programmers.

- **Simple design** requires programmers to adopt the simplest design possible and avoid complexity during implementation of user stories.

- **Test-driven development** provides mechanisms for the programmers write unit tests before implementing user stories in order to achieve a desired quality for the features.

- **Refactoring** the written code while maintaining the behaviour of the features is important to develop reliable and robust software products.

- **Pair programming** refers to a team of one driver and one navigator working together at one workstation in order to develop a piece of code.
• **Continuous integration** involves building of a software product through developing small features on a regular basis and integrating them with the current product several times a day, after each feature is completed.

• **Collective code ownership** implies that the development teams own the written code. Therefore, every programmer could improve any piece of code within the project code repository at any time, if necessary.

• **Customer tests** are user-acceptance tests which customers design and implement to validate the correctness of the features.

• **Sustainable pace**. No one can work a second consecutive week of overtime. Even isolated overtime used too frequently is a sign of deeper problems that must be addressed.

• **Whole team** refers to the project team which is responsible for the success of a project. While customers are integral part of a project team, the members of the project team work together to achieve the project goals.

• **Coding standards** allow programmers to maintain a common style for writing code for a project in order to facilitate communication between them.

### 2.1.4 Agile Teams

Agile software development focuses on the talents and skills of individuals and molds processes to specific people and teams [53]. Communication, collaboration and coordination between developers, customers, managers and other stakeholders are utmost important for the success of a software development project [212]. Beck [23] suggests a team in Agile software development comprises “a variety of people who work together in interlinking ways to make a project more effective”.

2.1. AGILE SOFTWARE DEVELOPMENT

While there is a large number of definitions for “team” from different perspectives, some definitions with similar connotation follows. Several classic definitions focus on the importance of individuals working together. For example, Francis and Young [97] define a team as “an energetic group of people who are committed to achieving common objectives, who work well together and enjoy doing so, and who produce high-quality results”, whereas Dyer [77] defines a team as “a social entity comprising members with high task interdependency, shared understanding, and shared values, working together to achieve common goals”. A number of definitions focus on what individuals in a team do. For example, Marschak defines a team as “a group of individuals, each of whom capable of taking decisions about different aspects of an involvement but receive a common reward as the joint result of all those decisions” [196], whereas Hollenbeck et al. [148] define a team as “a group that consist of interdependent individuals with different skills, who share the same consequences when a facing success or failure at the group level”. Belbin [27] defines a team, in a simple term, as “a congregation of individuals, each of whom has a clear role, which is known to one another”. A role within a team describes the pattern of behaviour in which one team member interacts with another where his or her performance serves to facilitate the progress of a team as a whole [27].

In a recent study on teamwork, Salas et al. [257] define a team as “a distinguishable set of two or more people who interact dynamically, interdependently, and adaptively towards a common and valued goal / objective / mission”. Similarly, Adair [4] defines a team as “a group in which the individuals share a common aim and in which the jobs and skills of each member fit in with those of the others”. In this thesis, with the software development perspective, we define a project team as “a group of committed, interdependent individuals with complementary skills who work together in order to complete a software development project”. 

In team development theory, Tuckman [301] recognises the fact that teams do not start off fully-formed and fully-functioning. He found teams develop from their creation as mere groups of individuals, to cohesive, task-focused teams, through four clearly defined stages: “forming”, “storming”, “norming” and “performing”. Tuckman later added a fifth phase “adjourning and transforming” to cover the finishing of a task [102]. Figure 2.5 illustrates the development stages of a team.

![Development stages of a team](image)

Figure 2.5: Development stages of a team (adapted from Tucker [301])

In the forming stage, the senior managers and key stakeholders provide clear guidelines, directions, and clear goals and objectives for the individuals within a team in order for them to know his/her roles and responsibilities [102]. In the storming stage, project managers acts as a coach in order to develop good understanding between members of the team [102]. In the norming stage, the roles and responsibilities of every team member are established and the team members develop solid cohesion between them [102]. Finally, in the performing stage, the teams strive to achieve the goals of the projects. Team members adopt a positive approach to resolve conflicts that might arise and develop mutual respect, shared understanding, and a shared vision [102]. A team can revert to the forming stage when members leave or a new member joins, but a successful team will develop cohesion and become fully-functioning within a short time [102].
2.1. AGILE SOFTWARE DEVELOPMENT

Drexler et al. [70] define the team performance model for understanding the path that team members take together to reach high performance. This model summarizes the basic dynamics of teams and involves seven stages: orientation, trust building, goal clarification, commitment, implementation, performance, and renewal. A critical part of the team performance is the development of team relationships among members of a team in order to provide a foundation for trust and commitment [306]. Figure 2.6 illustrates the team performance model.

In order to define what an Agile team is, however, first we need to understand what agility is. There exist a number of definitions for “agility” primarily based on the principles behind the Agile Manifesto [79]. Highsmith [131] define agility as “the ability to both create and respond to change in order to profit in a turbulent business environment”. Ericksson et al. [82] define agility as “the means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user-requirements, accelerated project deadlines and the like”. Kruchten [175] define agility as “the ability of an organisation to react to change in its environment faster than the rate of these changes”.

Figure 2.6: Team Performance Model (adapted from Drexler et al. [70])
These definitions imply agility is an ability of an organisation or a team to create innovative changes and to react quickly and effectively to both anticipated and unanticipated changes in the business environment in order to provide business value to customer. In this thesis, we define agility as “the ability of a team to embrace changes at any stage of a project in order to deliver working software that provides business value to customer”.

We consolidated our definitions of team and agility with the concepts of teams in Scrum and XP – Scrum Team and Whole Team – in order to define what an Agile team is. In this study, we define an Agile team as “a cross-functional, self-organising, high-performance team that carries out software development activities in line with the principles behind the Agile Manifesto in order to provide business value to customer”. A detailed description of what it takes for a team to be a cross-functional, self-organising and high-performance team follows.

In order to be a cross-functional team, the team comprises everybody involved in the development of a project, including people from the business area, project management, development, testing, quality assurance, and technical support. In particular, customer is an integral part of the Agile team. A cross-functional team as a whole has all required skills and knowledge, and every member of the team is willing share skills and knowledge with one another [147, 221]. A wealth of studies [43, 42, 79, 292] point out Agile methods recommend the use of cross-functional teams in order to promote direct communication, facilitate close collaboration and foster knowledge sharing within the team. Highsmith [132] argues cross-functional teams are capable of producing high quality deliverables throughout a project. Hossain, Babar and Paik [152] reported members of a cross-functional, distributed team attend and participate in every Scrum meeting in order to bridge spatial, temporal and socio-cultural distances (s 2.2.2).
In order to be a self-organising team, the members of the team manage their own workload, shift work among themselves based on need and best fit, and participate in team decision making while typically performing highly interdependent software development activities [129, 137]. A wealth of studies [13, 53, 47, 80, 96, 137, 273] point out “self-organising” is an important characteristic of a successful Agile team. Moe, Dingsøyr and Dybå [214] argue a self-organising team is in fact an “autonomous team” or an “empowered team”, in which a team as a whole has the autonomous to determine how and when to complete the tasks assigned for a sprint or a project. In general, self-organising is seen as the “first design principle” for an innovative and collaborative team [210], whereas in Agile software development, a self-organising team captures the “spirit of Agile values and Agile principles” through a focus on human and social aspects, trust and mutual respect [141].

In order to be a high-performance team, the team is highly cohesive and every member of the team focuses on the success of the project and strives to provide business value to customer [169]. Kur [177] argues a high-performance team “frequently outperforms other teams that produce similar products and services under similar conditions and constraints” because it constantly attempts to satisfy the customer. Kur [177] further describes a high-performance team is “purposeful, social, human-oriented, technical and systematical”. Castka et al. [40] suggest a high-performance team is a cross-functional, highly principle-driven, process-oriented team that provides value to the stakeholders and the organisation. Ramesh et al. [246] reported team cohesion contributes to create high-performance teams in Agile software development. Sutherland et al. [294] argue Agile methods provide avenues for average teams to “self-organise into high-performance teams”. To summarise, an Agile team is a strategy to success even in complex software development projects.
2.2 Distributed Teams

In this section, we describe distributed teams and the temporal, spatial, and cross-cultural distances.

2.2.1 Varieties of Team

There are varieties of software development teams that reflect how individuals are situated in a team while working together on a project. A co-located team comprises individuals who work together in close proximity with one another [51]. A dispersed team comprises individuals who work from different places while collaborating on the common software projects [36, 79]. A multi-site team is a relatively self-contained software development team that works on sub-system or relatively well-decoupled part of the whole project [51, 79]. A virtual team comprises independent individuals who work across space, time and organisational boundaries with link strengthened by webs of communication technologies [192, 79]. A global team comprises team members who are geographically dispersed and who may rarely meet, if at all, during the course of the project [38, 79].

Lipnack and Stamps [192] describe four varieties of team with respect to the space, time and organisation dimensions. The space and time are treated as a single inter-related phenomena because distance in space, no matter how short the distance is, takes time to cross [192]. A co-located team comprises individuals who work in the same space and time, on interdependent tasks; a co-located cross organisational team comprises individuals from different organisations who work together in the same space and time; a distributed team comprises individuals from the same organisation who work in different places either interdependently or separately; and a distributed cross organisational team comprises individuals from different organisations who work in different places. The teams that carry out interdependent tasks clearly have a spacetime distance problem to solve in order to work together. Figure 2.7 shows the varieties of teams.
Carmel and Abbott [38] argue there are key questions that should be clarified when discussing distributed software development. In particular, who or what is distributed and at what level? It is crucial to clarify whether people or the artifacts are distributed, and whether people are dispersed individually or dispersed in groups. With distributed software development, teams are not physically co-located and, therefore, team members cannot see or speak in person on a regular basis. Distributed software development ranges from team members being dispersed over adjacent buildings to being dispersed over different continents [242].

Global software development is the special case of distributed software development in which the dispersion of the team extends across national boundaries [185, 243]. In general, global software development refers to sourcing of software development work across different geographical sites and normally across different time zones and cultures [61, 79]. While there are significant differences in the approach to sourcing, global software development includes both outsourcing and offshoring.
Outsourcing is sourcing software development work to sub-contractors to other individuals or software companies [79, 167], whereas offshoring (both farshoring and nearshoring) is sourcing software development work to a different country, often one in the developing world [283]. Farshoring and nearshoring are seen as contrasts: a farshore destination represents a distant country with many hours to travel, many time zones away, and is seen to represent a very different culture; whereas a nearshore destination is associated with relatively easy travel, similar time zones, and closeness in culture and language [38].

In this thesis, we define a distributed team as “a project team from the same organisation where at least one member of the team is physically distant from the others in different countries, across different continents”. We investigated how distributed teams in Agile software development carried out software development projects while facing spatial, temporal and socio-cultural distances.

### 2.2.2 Global Distance

The term ‘global’ in global software development describe the software development efforts across geographical sites, and often across different time-zones and cultures. Hence, global distance implies the spatial distance, temporal distance, and socio-cultural distance between the members of a distributed team due to separation through space, time, and socio-culture [7, 8].

*Spatial distance* is a measure of the geographical dispersion, occurring when team members are dispersed across different sites [8, 181]. Spatial distance is often operationalised as the cost or effort required to exchange visits from one site to another. While spatial distance removes the ability for face-to-face communication, team members face difficulties to develop cohesion among team members [154].
Temporal distance is a measure of the temporal dispersion, occurring when team members wishing to interact [8, 181]. Though temporal distance is often caused by time-zone differences across sites, but it can also be caused by time shifting work patterns. Temporal distance reduces opportunities for synchronous communication, which translates into delayed response and less collaboration [154].

Socio-cultural distance is a measure of the effort required by team members to understand the national and organisational cultures across sites [8, 181]. Socio-cultural distance describes the degree to which the cultural backgrounds of the team members differ, and this can cause cross-cultural communication challenges.

2.2.3 Why Go Distributed?

There is a wealth of literature [37, 180, 290, 303] on why software companies are venturing into distributed software development. In general, the need to capitalise on the global talent and to expand business in a global market have accelerated this trend. In his seminal book, Erran Carmel [37] lists the six main ‘catalyst’ factors, or potential benefits, which have driven to distributed software development:

1. Access the most talented developers: The quality of software developers is the most important factor for the success of a software development project. Hence, hiring the most talented software developers throughout the world, regardless of their geographical location, is the most sensible approach for organisations that want to deploy market-winning software products.

2. Reduce development costs: Software companies in high-cost countries try to reduce development costs by outsourcing development work to software developers in low-cost countries (e.g., India, China, Brazil, and East Europe).
3. **Mergers and acquisitions:** The global demand for software products and services created a need for mergers and acquisitions, as software companies strived to penetrate new markets and adjust or complement existing product lines. As a result, software teams across the globe are forced to collaborate on common projects as an overall global team.

4. **Position as global organisations:** Software companies continue to position themselves as ‘global players’ to increase business opportunities with other global organisations that prefer comprehensive software suppliers for all their global subsidiaries rather than an heterogeneous network of vendors in different countries.

5. **Increase proximity to the market:** The business advantages of proximity to the market includes knowledge of customers and local conditions, such as localisation, customisation, and after-sale services, as well as the goodwill engendered by local investments such as a favorable tax treatment from governments.

6. **Reduce time to market:** With software developers scattered across global sites, distributed software development fosters ‘round-the-clock’, or ‘follow-the-sun’, development, which has the potential to permit the reduction of development cycles by increasing the amount of time in a day that software is being developed.

There is a large number of studies that discuss the potential challenges in distributed software development. Though, if given a choice, almost all project teams would sit together in a single room with the customer representative or product owner to maximise collaboration and improve communication, in reality there are forces at play that lead to distributed software development. Next, we discuss the key concerns of distributed teams based on the existing literature.
2.3 Concerns of Distributed Teams

In this section, we describe several concerns of distributed teams in the light of existing literature. A large number of studies point out the members of a distributed team are primarily concerned about how to promote communication, build trust, manage knowledge, and bridge cultural differences with one another in the team.

2.3.1 Promoting Communication

Communication is important for software development teams to facilitate knowledge transfer rapidly amongst team members, to allow team members to understand requirements from customers, and to help team members perform development activities efficiently [126, 240]. Communication provides avenues for team members to state ideas and opinions, listen to understand concerns of others in the team, and provide suggestions to the concerns raised in the team [164, 169]. Several studies [40, 83, 89] suggest the effectiveness of communication between members of a team determines the success or failure of the project.

Distributed teams, however, communicate over time and space through technology mediated communication such as e-mail, instant messaging, and video conferencing [7, 89, 185]. Non-verbal communication such as facial expressions and hand gestures that are often missing in technology mediated communication can decrease the awareness of team member interactions [89]. Several studies [126, 170, 171, 213, 243], however, point out that communication is the main challenge for a distributed team in which team members are scattered across different geographic locations, and often across several time zones.
A wealth of studies [126, 170, 171, 213, 221] point out that the success of Agile software development depends significantly on practices characterised by face-to-face communication within the team such as daily Scrum and retrospectives meeting. XP [23] states communication as one of the five driving values where everyone in the team ideally communicates face-to-face on a daily basis. XP primary practices such as Sit together, Whole team, Informative workspace, and Pair programming encourage communication channels to remain open at all times [23]. Further, face-to-face communication is viewed as the main enabler of collaboration between team members [23]. Face-to-face communication often solves the collaboration problems that arise within a team because conversation facilitates knowledge sharing and helps to develop cohesion between team members [23, 272].

Agile methods emphasise the importance of interaction between individuals in a team, and the Agile principle states face-to-face conversation as the most efficient and effective method of conveying information to and within a development team. Through there exist literature which discusses communication challenges in global software development, however, little is known how distributed teams in Agile software development promote synchronous communication across spatial, temporal, and socio-cultural distances.

### 2.3.2 Building Trust

Trust has been studied in many fields such as organisational management [202], psychology [248], computer science [197], social science [98] and medical [120].

Mayer et al. [202] define trust as the “willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party”.
2.3. CONCERNS OF DISTRIBUTED TEAMS

Based on Mayer’s [202] definition of trust, Doney et al. [68] define trust as “a willingness to rely on another party and to take action in circumstances where such action makes one vulnerable to the other party”. With expectations of a target’s trustworthiness drive a trustor’s behaviour, trustors engage in one or more cognitive processes in order to determine whether or not targets are trustworthy. In this thesis, we use Doney et al.’s [68] definition of trust because it includes both the belief and behavioural components of trust, and it also incorporates the notion that both trustor and target are necessary for trust to be present.

Trust plays an important role in determining the success or failure of software development projects. Hung et al. [157] points out the existence of high initial trust among team members in virtual teams although traditional models of trust have seen trust as being created as a result of a long history of interaction between individuals. They argue that individuals in a project team form trust attitudes through three distinct routes: peripheral, central, and habitual. In the initial stage of a relationship, members of a virtual team rely on peripheral cues such as third party information, social categories, roles and rules because they lack information about one other in order to form trust. Once individuals have shared history and knowledge of the other individual, they use the central route, which involves the assessment of the other individual’s ability, benevolence and integrity. While ability refers to the aptitude and skills that enable an individual to be perceived as competent by team members, benevolence refers to the extent to which an individual is believed to be willing to do good to team members beyond personal profit motive, and integrity refers to the extent to which an individual is believed to adhere to a set of principles thought to make the individual dependable and reliable [202]. Finally, after long periods of shared history in which the individuals develop a habitual pattern of trust, they simply enact prior trust attitudes via the habitual route.
Several studies [17, 171, 204, 216, 246, 303, 319] have discussed the importance of trust for Agile teams in particular. The self-organising nature of Agile teams has increased the importance of trust among team members [137, 273]. One Agile principle suggests managers must trust the Agile teams to deliver business values to customers [203]. Similarly, teams members must trust that individuals in the team are competent, knowledgeable, and collaborative to design, develop, and deliver valuable software [204, 221]. Despite the emphasis on the importance of trust, little is known how distributed teams in Agile software development build trust with one another across spatial, temporal and socio-cultural distances.

2.3.3 Managing Knowledge

One of the major reasons for software companies to venture into distributed software development is to benefit from the global knowledge resources [127, 185]. Self-organising Agile teams need to leverage knowledge to develop valuable software within time and budget but with a higher quality while embracing changes throughout the software development process [2, 276]. Though several studies [62, 160, 315] have defined knowledge, the following is one of the most common definition:

"Knowledge is a fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of the knowers. In organisations, it often becomes imbedded not only in documents or repositories but also in organisational routines, processes, practices, and norms." [62]

Knowledge is divided into two types: *tacit* and *explicit* [241]. *Tacit knowledge* is the action-oriented or “know how” knowledge that guides human behaviour but it cannot be openly expressed, whereas *explicit knowledge* is the academic knowledge or “know what” knowledge that can represented in written or verbal forms [281].
A wealth of literature points out the most valuable knowledge within an organisation is essentially tacit. This tacit knowledge is embodied in people in the form of experience and skills, and in the processes and products that people create. Nonaka and Takeuchi [225], writing about knowledge creating companies, describe that “although we use the term ‘organisational’ knowledge creation, the organisation cannot create knowledge on its own without the initiative of the individual and the interaction that takes place within the group”. This means that organisation create knowledge from the tacit knowledge of individuals in the organisation.

Knowledge Management (KM) is “a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company’s knowledge” [62]. KM describes the processes through which organisations manage tacit and explicit knowledge held within the individuals of the organisations to achieve competitive advantage [9, 262]. The approach to knowledge management often assumes that the knowledge within an organisation will largely consist of tacit knowledge that remains in the heads of individuals in the organisation. Wilson [315] suggests that “whenever we wish to express what we know, we can only do so by uttering messages of one kind or another – oral, written, graphic, gestural or even through body language”. Davenport and Prusak [62] explain that “knowledge exists within people, part and parcel of human complexity and unpredictability”, and therefore knowledge transmitted outside of the mind becomes mere information for the receiver. From this perspective, we understand that although organisations can offer tools, techniques, processes, and procedures, but managing knowledge still will remain the responsibility of the individual.

There is a wealth of literature on KM from different fields such as organisational theory, psychology, sociology, information system, and computer science. There are several different KM models for researchers and practitioners to generalise, communicate, and apply the findings [260]. Within a KM model, there are four knowledge processes: knowledge
generation (creation and acquisition), knowledge codification (storing), knowledge transfer (sharing), and knowledge application (use) [9, 62, 286]. Knowledge generation is creating innovation and opportunities for resolution of problems, and acquiring knowledge from external sources [62]. Knowledge codification is translation of tacit knowledge into explicit knowledge in written or verbal forms for storage in repositories [62]. Knowledge transfer is sharing of knowledge between individuals within an organisation [62]. Finally, knowledge application is using knowledge to gain the competitive advantage [9, 62].

The concept of knowledge management is important in software development, particularly in Agile software development. Co-located Agile teams emphasise face-to-face interaction and frequent communication with team members, in order to gather, store, share and use invaluable project-specific knowledge that is imperative for the success of software development projects [53, 208, 221]. Hence, Agile teams must understand how to manage knowledge on business domain, software development processes, tools, and technologies [42, 215]. Several studies, such as Boden et al. [31] and Holz and Maurer [151], point out that managing knowledge is difficult for distributed teams due to spatial, temporal, and sociocultural distances. Little is known how distributed teams manage knowledge in Agile software development.

2.3.4 Bridging Cultural Differences

Over the years, researchers have offered many different definitions of culture. Indeed, 60 years ago Kroeber et al. [173] had already identified 164 definitions of culture. It is therefore pointless to attempt to make sense and order out of such divergent views; and one may therefore feel free to choose a convenient definition according to one’s particular needs and sensitivities.
In general, culture can be viewed from two wide perspectives: national culture and organisational culture. National culture encompasses ethnic, racial, gender, and other demographic characteristics whereas organisational culture encompasses work groups with which an individual may associate such as teams, departments, or organisations [56, 60]. Language, which is a set of common sounds, symbols and gestures, is closely tied to the culture [168]. In fact, body language such as facial expressions and hand gestures is considered more meaningful than spoken language [89].

According to Hofstede [143, 144], culture is “the collective programming of the mind which distinguishes the members of one category of people from another”. Hofstede [143, 144] further describes the “category of people” can be a nation, region, or ethnic group (national culture), women versus men (gender culture), old versus young (age group and generation culture), a social class, a profession or occupation (occupational culture), a type of business, a work organisation or part of it (organisational culture), or even a family. Spencer-Oatey [284] defines culture as “a fuzzy set of attitudes, beliefs, behavioural norms, and basic assumptions and values that are shared by a group of people, and that influence each member’s behaviour and his/her interpretations of the ‘meaning’ of other people’s behaviour”. Clark [48] defines culture “as a distinctive, enduring pattern of behavior and/or personality characteristics”. The definition of culture, in general, includes some notion of shared values, beliefs, expectations, customs, jargons and rituals.

Hofstede [142] found differences in national cultures vary substantially along four dimensions: uncertainty avoidance, individuality, tolerance of power distance, and masculinity-feminine. The power distance dimension is defined in terms of the prevailing norms of inequality within a culture [142]. Individualism-collectivism refers to the extent to which the
identity of members of a given culture is shaped primarily by personal choices and achievements or by the groups to which they belong [142]. Masculinity-femininity corresponds to a “tough-tender” dimension [142]. Whereas in masculine cultures, values such as competition, success and performance are emphasised, in feminine cultures, values such as social relationships, quality of life, and care of the weak are emphasised instead. The fourth dimension, uncertainty avoidance, alludes to the degree to which members of a culture are uncomfortable with uncertainties in life [142]. Societies high on this dimension prefer structured rather than unstructured situations, where there are clear guidelines for behaviour.

In spite of the growing interest in organisational culture among behavioural scientists and practitioners, no strong consensus has been formed about a definition of the term. Hofstede et al. [145] describe organisational culture is a phenomenon of a different order from national culture because membership of an organisation is usually partial and voluntary, whereas the membership of a nation is indeed permanent and involuntary. Trompenaars and Hampden-Turner [300] investigate cultural impact at workplace based on Hofstede’s dimensional model. Trompenaars and Hampden-Turner [300] define organisational culture as “the way in which a group of people solves problems and resolves dilemmas”. Deshpande and Webster [64] define organisational culture as “the pattern of shared values and beliefs that help individuals understand organisational functioning and thus provide them norms for behaviour in the organisation”. Schein [261] defines organisational culture as “a standard set of basic suppositions invented, discovered or developed by the group when learning to face problems of external adaptation and internal integration”. In general, organisational culture in software companies includes the culture of systems development, such as the use of software processes and project management practices.
2.4. DISCUSSION

Culture is indeed responsible for the behavioural patterns of individuals in a group because culture affects the way individuals think, act, and communicate with others [3, 316]. In this thesis, we adopted Doney et al.’s [68] definition of culture as “a system of values and norms that are shared among a group of people and that when taken together constitute a design for living.” In this definition, we view culture from the national and organisational perspectives, in which culture is not a characteristic of individuals, nation states, or organisations but of a large number of people conditioned by similar background, education, and life experiences.

Cultural differences often arise in a distributed team that comprises individuals from different countries [3, 60]. Individuals from different cultures can react differently to the same situation due to the different beliefs, values, attitudes, or perception of priority [60, 316]. Therefore, cultural difference is one of the most important concern that should be addressed for distributed teams. Several studies [191, 292, 317] point out cultural differences among members of a distributed team affect software development projects. Little is known how distributed teams in Agile software development bridge cultural differences between team members across spatial, temporal, and socio-cultural distances.

2.4 Discussion

In this section, we discuss Agile software development from the research and practice perspectives. In particular, we examine the trends in research on Agile software development.

Early research on Agile software development explored the software development practices associated with Agile methods such as the test first development and pair programming [100, 223, 271]. Later, research examined the adoption of Agile methods to software development projects [276], and the transition from traditional software development practices to Agile software development practices [221].
In response to the growing number of software development teams using Agile methods, the academia is supporting Agile methods in software engineering and computer science curricula [207]. Williams, Kessler and Upchurch [313, 314], who have been evaluating pair programming model in computer science courses for many years, report that pair programming contributes to better teaching and learning of computer programming, and results in good quality of software developed by students. Melnik and Maurer [206, 209] report on student perceptions of software development using Agile methods that students were “very enthusiastic about core Agile practices”.

In recent years, the research on Agile has focused on the social aspects of the Agile teams. Whitworth and Biddle [311] conducted a Grounded Theory study to explore the socio-psychological experiences of Agile teams. This study suggests that there are strong social forces at play in Agile teams that underscore the Agile values. They suggest exploring and understanding the socio-psychological and cultural issues in software development teams is essential to self-organising Agile teams.

Martin, Biddle and Noble [198, 199] conducted a qualitative research study of XP customers. They explored the customer’s explicit responsibilities in the Agile software development projects. They reported that customer not only provide project requirements and conduct acceptance testing, but customer must also shoulder a number of implicit responsibilities including liaison with external project stake-holders while maintaining the trust of both the development team and the business people. They found the customer have a pressured and stressful role, leading to issues of sustainability.
2.4. **DISCUSSION**

Hoda, Noble and Marshall [136, 137] conducted Grounded Theory research to explore the self-organising nature of the Agile teams. They identified six informal roles – Mentor, Co-ordinator, Translator, Champion, Promoter, and Terminator – that team members adopt in Agile software development. The researchers argue that self-organising team is one of the critical success factors on Agile projects, and the Agile team members adopt one or more of these six informal roles to facilitate their team’s self-organisation.

Ferreira, Noble and Biddle [86, 87] conducted Grounded Theory research to investigate role of interaction designers in Agile software development. They reported on how the interaction designers combined interaction design activities with Agile software development with a focus on the issue of interaction design being done “up-front” before commencing the actual software development.

The use of Agile methods in distributed software development is rapidly becoming common for software companies [153, 170]. Studies [153, 280] report the overall combination of Agile software development and distributed software development are not well understood. We further discuss several recent studies that report the challenges in adopting Agile methods to distributed software development.

Vax and Michaud [303] describe the potential challenges in distributed software development such as the inability to meet face-to-face on daily basis, language, and cultural barriers, and lack of trust. They explain that adopting Agile methods to distributed projects can be challenging but by constructing the right team with the right skills and expertise, and effectively leveraging tools and techniques, the teams can acquire tremendous benefits in terms of scalability, productivity, cost management, risk reduction and improved software quality.
Sureshchandra and Shrinivasavadhini [290] report their experience in handling several Agile software development projects in different domains. They analysed the performance of distributed teams on Agile projects in contrast to the non-Agile projects on their organisation. The analysis shows that distributed teams on Agile projects fared better than distributed teams on non-Agile projects in terms of schedule and overall productivity without negative impacts on effort and quality. The distributed teams on Agile projects provide customers with better ability to handle requirements volatility and reinforce that stakeholders can gain benefits of both distributed software development and Agile methods.

Hsieh et al. [156] examined the concept of culture and the potential impact of intercultural dynamics on global software development projects, and presented a descriptive conceptual framework for coordination between individuals and teams. They argue cultural differences create mismatch in expectations between remote sites as a result of communication delays and noise, differing language skills, and disconnects in knowledge.

Young and Terashima [319] recognise that having distributed Agile teams present many challenges that slowed down software development processes. They describe the strategies adopted by distributed Agile teams to overcome time zones problems and cultures differences. In order to deliver successful software releases, the distributed teams deliberately made conscious efforts to improve communication, built strong working relationships amongst the distributed team members, and ensured that the software architecture was suitable for all teams.

Korkala and Abrahamsson [170] recognise that distributed software development is increasingly becoming important for software companies. They explain that distributed software development is already burdened with several challenges, and Agile methods bring further challenges in
2.4. DISCUSSION

the form of their reliance on informal communication and volatile requirements. They describe that the high volatile requirements in Agile software development are managed through effective communication. Further, they recommend enabling and supporting direct communication between developers contributes to the success of Agile software development with distributed teams.

Ramesh, Cao, Mohan and Xu [246] investigated the challenges that arise from blending agility with distributed development by studying three organisation that have adapted their practice to support distributed development. They argue Agile practitioners need to carefully incorporate agility in distributed software development, and present practices that demonstrate how a balance between Agile and distributed approaches can help resolve challenges in communication, control, and trust across distributed teams.

Most of the literature pertaining to Agile software development with distributed teams presented here is revisited, and further literature is discussed in relation to the research findings as discussion sections at the end of chapters 4–8.
Chapter 3

Research Design

In this chapter, we provide a detailed description of the research design, present the research participants and projects, describe the data collection and analysis processes, and finally discuss the evaluation of the theory.

3.1 Research Paradigms

The notion ‘paradigm’ was introduced by Kuhn [176] to describe the entire constellation of beliefs, values, techniques, laws, and instrumentation shared by the members of scientific community which gives rise to, and serves as a model for, a "coherent tradition of scientific research". Kuhn [176] argues that all disciplinary research begins at a philosophical level that defines a scientific paradigm for the generation of knowledge. Kuhn [176] suggests that scientific paradigms provide different perceptions, understandings, and beliefs which guide the adoption of a suitable method to observe and measure research phenomena. Guba and Lincoln [118] define paradigms as the “basic belief systems based on ontological, epistemological, and methodological assumptions”. Ontology is the nature of social reality; epistemology is the assumption concerned with how researchers gain knowledge of reality; and methodology is the practice of how exactly researchers come to know that reality [118, 218, 227, 307].
Ontology is the starting point of all research, from which the epistemological and methodological positions of the researchers should logically follow. The paradigm boundaries can be established through realising the ontological, epistemological, and methodological stances of the researcher. While there are several guiding philosophical assumptions such as Myers [218] and Guba and Lincoln [118], we found Orlikowski and Baroudi [227] provide clear and concise descriptions of the underlying ontological, epistemological and methodological stances that are appropriate for this research study.

The scientific paradigms that have been used in computing disciplines include the Positivist, Interpretivist, Postpositivist, and Critical Theory [116, 218, 307]. Software engineering researchers use scientific paradigms to understand software processes and produce software engineering knowledge [35]. In the following sections, we provide a brief description of these paradigms in the light of the ontological, epistemological, and methodological stances.

3.1.1 Positivist

The positivist paradigm originates from a philosophy known as logical positivism or logical empiricism, which is based on rigid rules of logic and measurement, truth, absolute principles and prediction [307]. Positivist assumes the existence of objective, independent, and stable realities whose nature can be unproblematically apprehended, characterised, and measured [118, 227]. Positivist relies typically on deductive logic, emphasises researcher perspective, and establishes priori assumptions about the research phenomena [118, 227]. According to Orlikowski and Baroudi [227]:

“Positivist studies are premised on the existence of a priori fixed relationships within phenomena which are typically investigated with structured instrumentation. Such studies serve to test theory, in an attempt to increase predictive understanding of phenomena.”
3.1. RESEARCH PARADIGMS

Positivist research considers researchers as separate from the research phenomena that are subject to observation, and findings are based on precise observations that other researchers can repeat [94, 227]. Researchers have a passive, neutral role in positivist studies, and do not intervene in the phenomena being studied [227]. Positivist studies focus primarily on experimentation and hypothesis testing to increase the predictive understanding of the research phenomena, and knowledge is summarised in the form of time- and context-free generalisation, often in the form of cause-effect laws [118, 227].

3.1.2 Interpretivist

Interpretivist research assumes the existence of social contexts that create realities whose natures do not exist apart from humans, and hence attempts to understand how and why individuals act the way they do [227]. Interpretivist research relies on inductive logic, emphasises participant perspective, and rejects a priori assumption about research phenomena [239]. According to Orlikowski and Baroudi [227]:

“In interpretive studies assume that people create and associate their own subjective and intersubjective meanings as they interact with the world around them. Interpretive researchers thus attempt to understand phenomena through accessing the meanings that participants assign to them.”

Interpretivist research considers findings are fully dependent of researchers and bound in time and context, and knowledge creation occurs through social interaction between the researchers and individuals within area of study [218, 227, 239]. Interpretive researchers believe that social reality can only be interpreted, and hence cannot be apprehended, characterised, and measured in an objective manner [227]. Interpretive studies focus primarily on understanding the meaning of human experiences and action within the research phenomena.
3.1.3 Postpositivist

Postpositivist research assumes the existence of multiple constructed realities instead of a single objective reality. These realities can be apprehended, characterised, and measured only “reasonably closely” rather than in absolute sense as a result of human fallibility [118, 239].

Postpositivist research considers that findings from a study are subject to human interpretation, and therefore emphasises on verification of findings to demonstrate objectivity [233, 239]. Researchers strive to be as neutral as possible to discover findings that are independent from the researchers. Postpositivist studies focus primarily on experimentation and hypothesis testing with significant emphasis placed on context of the research phenomena [118, 239].

3.1.4 Critical Theory

Critical Theory research assumes the existence of social contexts that create realities that are historically constituted and can be iteratively reproduced by people despite the changes in their social and economic status [227]. Critical research attempts to be a social critique and highlights the oppositions, conflicts, and contradictions within the social group [218, 227]. According to Orlikowski and Baroudi [227]:

“Critical studies aim to critique status quo, through the exposure of what are believed to be deep-seated, structural contradictions within social systems, and thereby to transform these alienating and restrictive social conditions.”

Critical Theory research focus primarily on emancipation of political, social, historical and cultural practices through understanding the embedded barriers and hidden power imbalances in a community and disclosing this knowledge to the public in order to raise awareness of social problems [78, 307].
3.1. RESEARCH PARADIGMS

3.1.5 Selecting a research paradigm

This study aims to understand how distributed teams in Agile software development communicate, collaborate and coordinate in order to deliver working software to customer. We realise that we need to investigate not just the software development processes, but also the social processes within the teams in particular, and within the organisations in general. We recognise the relativist form and nature of reality, which is locally constituted (within the individuals, teams or organisations) and specific to context (distributed teams in Agile software development). This strongly suggests the appropriateness of an interpretive paradigm [227]. We thus carried out our research through the lens of interpretive paradigm:

- we emphasised participant perspectives as a way to investigate the main concerns of distributed teams in Agile software development;
- we aimed to develop a theory about Agile software development with distributed teams rather than to test an existing hypothesis;
- our findings were bound within the context studied;
- knowledge on the main concerns of distributed teams in Agile software development is primarily acquired through conversation or social interaction between the researcher and participants.

This study adopts an interpretive paradigm to investigate distributed teams in Agile software development to generate a theory based on data. The adoption of an interpretive paradigm indicates this study recognises the importance of human actions and interactions to software development practice and research. In this paradigm, the researcher plays an active role in collecting data, such as deciding what to investigate and what questions to ask progressively, as well as in analysing data, such as what concerns are key to resolve problem within the research phenomenon.
3.2 Research Approaches

In the next section, we provide a detailed description of the quantitative and qualitative research approaches, and subsequently present our motivation for adopting a qualitative approach.

Quantitative research studies natural phenomena through experimentation and hypothesis testing, and therefore measures and analyses causal relationships between variables within the phenomena being studied [30, 218, 239]. Hence, quantitative research collects numerical data from a representative sample and examines them usually through statistical methods [30, 195, 218]. The main objective of a quantitative study is to identify the dependent and independent variables, and eliminate inadequate variables, so that the initial hypothesis can be confirmed or discarded [30, 195].

Qualitative research studies social and cultural phenomena through social interaction with individuals involved in the phenomena being studied [218, 236, 268]. Hence, qualitative research involves the use of visual and textual data that are acquired from interviews, observations, and documents. The main objective of a qualitative study is to understand the research participants and the social and cultural contexts within which they act, so that the researcher can explicate the phenomena being studied [195, 218, 236].

A number of studies [78, 239, 307] suggest a paradigm implies a particular research approach. For example, the positivist paradigm is associated with the quantitative approach because positivist studies focus primarily on experimentation and hypothesis testing to increase the predictive understanding of the research phenomena, and knowledge is summarised in the form of time- and context-free generalisation, often in the form of cause-effect laws [118, 227, 239]. The interpretive and critical the-
ory paradigms are associated with the qualitative approach because interpretive and critical studies intend to understand, describe, and explain research phenomena through social interactions and communications between the researchers and individuals within the area of study [15, 236, 239]. The postpositivist paradigm is associated with a dualism that includes both research approaches because postpositivist studies focus primarily on experimentation and hypothesis testing with significant emphasis placed on context of the research phenomena [118, 227, 239].

We decided to adopt a qualitative approach for our research for two important reasons: (1) the primary focus of our research is to understand human thought and action in software development contexts, and (2) we found that qualitative data such as interviews and observations provide significant descriptions that are required to understand and explicate the concerns of distributed teams in Agile software development.

Qualitative research also lends itself to studies of complex software engineering issues for generating well-grounded knowledge in areas that are not well developed theoretically or empirically, and where the research phenomenon requires a rich explanation for developing constructs and hypothesis [75, 268].

3.3 Research Methods

In this section we briefly describe three major research methods: Case Study, Ethnography, and Grounded Theory, and we present our motivation for choosing Grounded Theory as our research method.
3.3.1 Case Study

A case study is defined as “an empirical inquiry that investigates a contemporary phenomenon within its real-life context, when the boundaries between phenomenon and context are not clearly evident, and in which multiple sources of evidence are used” [318]. Cases are selected accordingly based on the pre-formulated research questions [255]. Case studies are considered for research questions that require in-depth investigation for the purpose of gaining understanding and knowledge of a phenomenon within its context, space and time [218, 318]. Case studies have been used effectively to study social aspects of Agile teams [122, 172, 184, 229].

3.3.2 Ethnography

Ethnography literally means “a description of a community”, and is defined as the “art and science of describing a human group – its institutions, interpersonal behaviours, material productions, and beliefs” [15]. Ethnography is used for long duration studies, with large amounts of data acquired via interviews, observation, or document analysis [205, 255]. Ethnography researchers immerse themselves in the research context for several months if not years, documenting behaviours of individuals or specific events that take place in a social community for the purpose of building theory or hypothesis by describing and interpreting situations in the community [88, 234, 239]. Ethnography has been used effectively to study social aspects of Agile teams [46, 214, 250, 272].

3.3.3 Grounded Theory

Grounded Theory is an inductive research method that emphasises the generation of a theory derived from systematic and rigorous analysis of data [114]. Grounded Theory researchers gather data, particularly qualitative data from interviews and observations, and systematically discover
3.3. RESEARCH METHODS

a theory derived directly from the data [140, 199]. Grounded Theory is intended neither to answer a specific research question nor to test an existing hypothesis, but rather to focus on discovery of a substantive theory regarding the research phenomenon within its context [11, 41, 239]. Grounded Theory has been used effectively to study social aspects of Agile teams [87, 140, 158, 199, 246, 311].

3.3.4 Selecting a research method

The fundamental principle for choosing a particular research method is the need to match the method with the ontological, epistemological, and methodological beliefs of the researchers [227, 239]. While Case Study and Ethnography were suitable candidate methods for qualitative research, we did not choose Case Study and Ethnography for the following reasons:

- Case studies requires a well established research question that can only be formulated from reading extensive literature on the phenomenon being studied. We aim to understand the main concerns of distributed teams in Agile software development from social interaction with the participants but not from existing literature. Moreover, we did not aim to test an existing hypothesis. That is, our ontological and epistemological stance did not favour Case Study as a method for our research.

- Ethnography requires the researcher to spend an extended amount of time in the community being studied. The researcher need to become a participant-observer who balances the objective collection of data with subjective insights that results from ongoing interaction between the researcher and participants. In order to investigate distributed teams, ethnography may require the researchers to travel to different locations and immerse in the teams for a period of time. Due to financial and logistic constraints, it was not viable to adopt ethnography for our research.
CHAPTER 3. RESEARCH DESIGN

Grounded Theory, however, stands out as a suitable method for our research for the following reasons:

• Glaser [105, 106, 111] argues that Ground Theory is ontologically and epistemologically neutral. That is, Grounded Theory can adopt any epistemological stance appropriate to the data and the ontological stance of the researcher for a specific research [33, 104, 111]. Glaser [111] argues that “whether Grounded Theory takes on the mantle for the moment of Prepositivist, Positivist, Postpositivist, Postmodernism, Naturalism, Realism etc, will be dependent on its application to the type of data in a specific research”. In practical, we can use Grounded Theory for this interpretive, qualitative research.

• Grounded Theory is presented as a general method which can use all data, quantitative and qualitative [33, 105]. Glaser [109] argues “the data is what it is, and the researcher collects, codes and analyzes exactly what he has – whether baseline data, properline data, interpreted data or vague data. There is no such thing for Grounded Theory as bias data, or subjective or objective data, or misinterpreted data.” Therefore, we can conceptualise the main concerns of the participants, irrespective of the source of the data, type of the data, and nature of the data.

• Grounded Theory is suitable for areas that are under-explored or where a new perspective might be beneficial [263, 5, 6], and literature on Agile software development with distributed teams is still scarce [73, 170, 230, 297].

• Grounded Theory allows researchers to study social interactions and behaviour of people [114], and Agile methods focus on people and their interactions in software development teams [96, 266].

• There is a supportive research group and a successful history of using Grounded Theory for investigating social aspects of Agile teams within the Victoria University of Wellington [85, 134, 200].
3.4. ROLE OF THE RESEARCHER

In section 3.5 we provide a detailed description of Grounded Theory and its application in this research, specifically the data collection and analysis processes, theory generation and evaluation. In the next section, we present the evaluation of self in the role of the researcher.

3.4 Role of the Researcher

In an interpretive, qualitative research, a researcher is an integral part of the research process. While the researcher is attempting the difficult task of understanding the research phenomena through the interpretations of the participants, it is important that the researchers have a view of their own role in this complex human process [34]. We thus provide a brief background of the researcher for the following reasons:

• researchers are important aspects of interpretive, qualitative research, where a researcher is viewed as an instrument through which data collection and analysis are conducted [234, 236]. The quality of data is highly dependent on researcher’s skill in interviewing, observing and gathering physical evidence;

• Grounded Theory demands a researcher who is “creative, continuously questioning, reasoning, and making sense of the connections between categories” so that the researcher can conceptualise and formulate a theory as it emerges from the data [25, 114];

• researcher bias can be addressed with self-reflection early in the research process where the a researcher describes personal experience and self-disclose personal assumption and beliefs that shape the interpretive, qualitative research [58, 103].
Though I have a vast experience teaching software engineering courses at universities and working on software development projects, I was careful not to let this experience cloud our research. I approached interviews with participants and observations of software development processes with an open mind. I was also careful not to read extensive literature from the same substantive area before commencing the research study. I have increased my understanding of the application of Grounded Theory through discussions with supervisors and mentors within the VUW, and reading professional literature from well-respected Grounded Theory research from different disciplines (e.g. [16, 18, 71, 123, 201, 263]). I, however, gained a deeper knowledge in the substantive area during interaction with participants through interviews and observations.

In general, there are two different roles which can be identified in an interpretive, qualitative research: “outside researcher” and “involved researcher” [304]. In the “outside researcher” role, a researcher carries out a study mainly through formal interviews, with no direct involvement in action in the research phenomena, whereas in the “involved researcher” role, a researcher is directly involved in action in the research phenomena as a participant observer or action researcher. In this study, I played the role of an “outside researcher”, where I was neither involved with the participants in the software development projects nor working in the same organisation with the participants.

The interpersonal skills of the researcher such as communication and listening skills are critical to entering the natural settings, data collection, and interpreting meaning [34]. In general, interviewees are normally willing to talk about their work and life experiences, with reasonable openness and honesty, provided that they trust the researcher’s statements on confidentiality and understand the researcher’s research agenda [304]. Based on prior experience in conducting in-depth interviews with participants from different cultural backgrounds and carrying out qualitative research studies, I knew it was important to develop solid professional relation-
ships with participants. I maintained communication with them throughout the research study for the purpose of providing them with all relevant information about this research; in particular about conducting interviews and sharing the findings. During the research, participants were assured confidentiality through the use of pseudonyms in the reporting of data.

3.5 Grounded Theory

In this section, we provide a brief history of Grounded Theory, discuss in detail the application of Grounded Theory in our research, and finally describe how we plan to evaluate and validate the emergent theory.

3.5.1 Brief History

Grounded Theory (GT) is defined as "a general methodology of analysis linked with data collection that uses a systematically applied set of methods to generate an inductive theory about a substantive area" [108]. Grounded Theory was jointly developed and first articulated by Barney G. Glaser and Anselm L. Strauss [114]. In the 1960s, Glaser and Strauss began a collaborative study in medical sociology to investigate the handling of dying patients in hospitals [113]. To describe how that study was conducted, Glaser and Strauss published their landmark book “Discovery of Grounded Theory” [114] in 1967, which laid the foundation of Grounded Theory. In that book, Glaser and Strauss [114] said that graduate students in the social sciences were being trained to confirm the ideas of early theorists but were not being encouraged to generate theory themselves. This one-sidedness thus was an interruption in the flow of ideas and a failure to appreciate the complexity and diversity of social life [182]. Glaser and Strauss [114] also said that the aim of that book was for “closing the embarrassing gap between theory and research” and “improving social scientists capacity for generating theory”.

The goal of Grounded Theory is “to generate a theory by systematic collection and analysis of data” [108]. In generating a substantive theory, a Grounded Theory researcher systematically collects data and rigorously analyses them at increasing levels of abstraction to uncover the main concern of the research participants and understand how they resolve it. The emergent theory is fundamentally a conceptual explication for the pattern of behaviour of the participants seen in the substantive area [11, 108, 109].

Grounded Theory developed into two versions – ‘Glaserian GT’ and ‘Straussian GT’ – due to the differences between Glaser and Strauss [41]. Glaser [108] argues Straussian GT is no longer Grounded Theory that generates theories which are grounded in data, but rather “full conceptual descriptions” that results from “forcing preconceived notions on data”. In this research, we have adopted the Glaserian GT for the following reasons:

- Glaserian GT generates theories that are emergent from empirical data rather than from inferences or existing theories, whereas Straussian GT forces categories and their properties into a uniform, pre-defined causal structure [186]. We were interested to investigate the research phenomenon and develop a theory that emerges from the empirical data rather than inferences or existing theories.

- Glaserian GT relies on the researcher’s conceptualising skills and theoretical sensitivity to generate theories, whereas Straussian GT relies on deeply structured process for constructing and linking categories [186]. We have complete trust in our ability to conceptualise data in order to develop a theory which reflects the main concerns of participants in the research phenomenon.
3.5. GROUNDED THEORY

- Glaserian GT suggests that the core category can be any kind of theoretical code, for example, a process, a condition, or a consequence that emerge from the data, whereas Straussian GT forces a “process” nature onto the underlying concepts [103, 288]. We kept an open mind with regard to how we will present the grounded theory as long as the emergent theoretical codes best fit our theory.

- There are more Glaserian GT resources available in terms of books, web sites, and particularly mentors [6]. The primary researcher continually had access to supervisors, mentors such as Dr. George Allan and Dr. Hans Lehmann, and co-researcher such as Dr. Rashina Hoda to discuss how to use the Glaserian GT in software engineering research.

Therefore, the term ‘Grounded Theory’ in the rest of this thesis refers to the Glaserian version of Grounded Theory (which is also known as classic Grounded Theory), and the data analysis steps follow the guidelines suggested by Glaser [107, 103, 108, 109, 110, 111, 112] and the original Grounded Theory [114].

3.5.2 Theoretical Sensitivity

A Grounded Theory research requires researchers to be creative and develop “theoretical sensitivity” to the subtleties of the data in order to conceptualise and formulate a theory as it emerges from the data [114]. Theoretical sensitivity is a quality of the researcher that combines interpersonal perceptiveness with conceptual thinking, and which is developed through professional and personal experience, combined with an in-depth understanding of literature during the research [103, 296]. Theoretical sensitivity allows the researcher to recognise what is important in the data, give them meaning during data analysis, and develop a theory that is grounded, theoretically rich and well integrated [302].
Glaser and Strauss [114] recommend that researchers start a Grounded Theory study with an “open mind” and Seidel and Kelle [269] explain that “an open mind does not mean an empty head”. Researchers are advised against reading extensive literature from the same substantive area before commencing the research study. Researchers are, however, strongly advised to read well-respected Grounded Theory research from different disciplines in order to increase their theoretical sensitivity of the application of Grounded Theory:

“The first step in gaining theoretical sensitivity is to enter the research setting with as few pre-determined ideas as possible – especially logically deducted, a prior hypothesis. In this posture, the analyst is able to remain sensitive to the data by being able to record events and detect happenings without first having them filtered through and squared with pre-existing hypothesis and biases.” [103]

According to Glaser and Strauss [114], potential theoretical sensitivity is lost when researchers read extensive literature on the same substantive area because the researchers are preoccupied with “testing, modifying, and seeing” the data from a preconceived angle.

3.5.3 Research Area

Glaser [114] suggests that a Grounded Theory study starts with a general area of interest rather than a specific research question for the following reasons:

- In order to design a research question, the researcher needs to read extensive literature in the same substantive area of the research, which in turn can affect the researcher’s theoretical sensitivity and ability to remain open to what is actually happening [103].
3.5. GROUNDED THEORY

- Defining research questions leads to preconceived ideas or hypothesis of the research phenomenon [108, 109]. The research problem should belong to the subjects (individuals, groups, or community), and therefore the problem should not be preconceived or forced [103].

- Grounded Theory is generally adopted for research with problems that are not well defined at the beginning, but incidentally discovered and focused through the researcher’s theoretical sensitivity and openness to the emerging theory [25]. That is, the problem and its key concerns should emerge from the data in the initial stage of data analysis [108, 109].

Therefore, we started out our research with a general area of interest – *Agile software development with distributed teams* – and not with a specific research question. As we attempted to avoid forcing any preconceived ideas on the research study, we were certain that the problem and its concerns will emerge in the initial stages of data analysis.

3.5.4 Use of the Literature

A dictum in Grounded Theory is that “there is a need not to review any of the literature in the substantive area under study” [108]. Glaser [103, 108, 110] wrote extensively about the danger of reading literature in the substantive area before commencing a research study. The reason is to prevent generating a focus from the literature rather than from the data if the researcher is “too imbued with concepts from the literature” [103]. Therefore, Glaser insists that reading literature before commencing the research study “violates the basic premise of Grounded Theory” [106]. Glaser explains that “it is hard enough to generate ones own ideas without the rich derailment provided by the literature” [103]. Hence, only by “not knowing” can a researcher be sure that the concerns emerging from data are an actual reflection of that data rather than preconception based on the literature [108].
Suddaby [289] argues that it is impossible for Grounded Theory researchers to ignore prior knowledge, but it is certainly possible for them to avoid formal reviews of relevant literature. Further, Suddaby [289] explains that the real danger of prior knowledge in a Grounded Theory research is that it will force the researcher into testing hypotheses, either overtly or unconsciously. Heath and Cowley [123] explain that hardly anyone commences a research study completely free from the influence of past experience and reading. It is, however, important for a novice researcher to recognise that a pre-study literature review can lead to pre-judgement and closing of ideas; result in the researcher focusing the research problem on areas that the literature suggests, rather than on those the data reveal to be important; and change the direction of the emerging theory to fit the literature [123, 128, 289]. In fact, Glaser [109] discusses near misses in discovering new theory whereby as the theory begins to emerge, the researcher reads closely relevant literature and its impact bends the emerging theory from its true direction.

Glaser [110, 111] explains even though a pre-study literature review is depreciated, the Grounded Theory is, however, not an excuse to ignore the literature. Glaser [110] suggests that “all is data”, meaning just that: “exactly what is going on in the research area is the data, whatever the source, whether interview, observations, documents”. Grounded Theory treats the literature as another source of data that can be integrated into the emerging theory through the data analysis process after the core category, its properties, and all other related categories have emerged [108, 106]. Glaser [103] states:

“We collect the data in the field first. Then start analyzing it and generating theory. When the theory seems sufficiently grounded and developed, then we review the literature in the field and relate the theory to it through integration of ideas.”
Since literature in Grounded Theory is perceived as another source of data, we decided to review the literature when the emerging concepts and categories were sufficiently grounded, and then integrate them into the emerging theory. In this thesis, we have provided a minor literature review in chapter 2 for the benefit of the readers. The major literature review, however, was conducted when emergent categories and their properties were sufficiently grounded and developed, particularly for the purpose of tying the emerging theory into existing literature. In chapters 4–9, the research findings are presented first, followed by discussion in the light of existing literature (s 4.2, s 5.2, s 6.2, s 7.3, s 8.3, and s 9.6).

3.6 Data Collection

In this section, we discuss interviews and observations as the data collection techniques for qualitative research in general, and describe how we recruited the research participants and conducted data collection through interviews and observations for this research study in particular.

3.6.1 Theoretical Sampling

In Grounded Theory, sampling occurs throughout the research study in order to facilitate the development of theory [114, 302]. The population under study is a set of concepts that constitute the phenomena rather than individuals experiencing the phenomena [6]. The initial decision for sampling is based on a general perspective and a general area of interest rather than preconceived theoretical ideas [25, 302]. After the emergence of the main concerns, sampling focuses on data collection for these main concerns, for the purpose of generating the conceptual hypotheses. Sampling in Grounded Theory thus begins with selective sampling and moves into theoretical sampling when concepts begin to emerge.
Selective sampling is the identification of populations and settings prior to data collection. In contrast, theoretical sampling is the process of data collection directed by emerging theory rather than by predetermined population dimensions [69]. Glaser [103] describes theoretical sampling occurs when the decisions on what data to collect and where to find them are determined based on the emerging theory:

“Theoretical sampling is the process of data collection for generating theory whereby the analyst jointly collects, codes and analyses his data, and decides what data to collect next and where to find them, in order to develop his theory as it emerges” [114].

Whereas the selective sampling is a tentative sampling strategy for beginning theory development, the theoretical sampling is the pivotal sampling strategy in Grounded Theory for developing a conceptually dense substantive theory. Researchers, however, must decide when to shift from selective to theoretical sampling, which is critical to the development of a conceptually-dense and complex theory. Figures 3.1 shows the sampling approaches for generating theory in a Grounded Theory research (adapted from Urquhart, Lehmann and Myers [302]).

Glaser [103] presents several recommendations for theoretical sampling, including staying open by changing interviewing styles, sites, or participants; follow up on recurring patterns in the data; and asking participants to give substantive information on concepts that seem central to the emerging theory. Grounded Theory research requires theoretical sampling to be continued until theoretical saturation [114] is reached – that is when no more new concepts or categories emerge from the data, and further data collection would be a waste of time:
3.6. DATA COLLECTION

"Saturation means that no additional data are being found to develop further the properties of a category." [103].

In this research, theoretical sampling guided us particularly in the selection of suitable participants who provided explanations on the emerging main concerns (s 3.6.2) and the evolution of interview questions that focused on the emerging concerns (s 3.6.3). Through theoretical sampling, we continued to develop a conceptually dense substantive theory until there was no more new constructs emerging from the data. In the following subsections, we discuss how we recruited participants and how we conducted interviews and observations in this research study.
3.6.2 Recruiting Participants

We initially carried out selective sampling based on three fundamental criteria in order to begin theory development. First, we decided to start out with participants from software companies in the USA. This is mainly because the USA is the home to a well-established software industry with significant number of Agile adoptions [246, 291]. Second, the participants must have worked with a distributed team in Agile software development, either in a technical, business or management role. Third, the participants comes from a distributed team which has been dispersed across different countries, in different continents (s 2.2.1).

We identified potential participants through profiles available in the LinkedIn communities of practice such as the Agile and Lean Software Development, the Agile Professional Network, and the Agile Project Management. Some of these individuals have been actively participating in open forums and regularly writing blogs about distributed teams. When we were somewhat certain that these individuals met the basic requirements to participate in this study, we sent out e-mails to them explaining our intention to carry out a research study on Agile software development with distributed teams. We provided them with detailed information about this study, and attached the relevant documents, such as Attachment A: Cover Letter for Research Participant, Attachment B: Information Sheet, Attachment C: Consent for Research Participation, and Attachment D: Interview Guide (see Appendix B). To our advantage, most of these individuals agreed to participate in this study. We set appointments for the interviews, which included the date, time and venue. The primary researcher personally travelled to the USA and conducted the interviews with them in August/September 2009. These initial participants were primarily Developers, Scrum Masters and Agile Coaches, including one Agile Manifesto signatory.
3.6. DATA COLLECTION

While analysing the initial data (s 3.7), we recognised the need to include participants from different positions such as Business Analyst and Quality Analyst in order to get a rounded perspective of the research phenomenon. We also recognised the need to include participants from India because a majority of the teams had team members distributed between the USA and India. Hence, we decided that the next destination for sampling must be India and we must include participants from different positions in order to develop a conceptually-dense substantive theory.

Using profiles in the communities of practice and the contacts established through the existing participants and software companies, we identified a number of potential participants from India. We followed the same process of writing e-mails to them and seeking their support to participate in this study. Once the appointments for the interviews were set, the primary researcher travelled to India to conduct the interviews with participants in May/June 2010. Ongoing analysis of data pointed out that now we must also include participants from Australia because several teams had team members distributed between India and Australia. We also recognised a need to include participants from different positions such as Product Owner and Senior Manager in order to get a rounded perspective of the research phenomenon. In September 2010, the primary researcher travelled to Australia to conduct interviews with participants. The primary researcher again travelled to India in February/March 2011 and the USA in August/September 2011 to conduct interviews with participants in order to strengthen the emerging theory.

In this study, all the interviews were conducted in the USA (Chicago, San Francisco, Salt Lake City, and Dallas), India (Bengaluru, Thiruvananthapuram, and Chennai), and Australia (Melbourne). Hence, the teams were primarily distributed across the USA, India, and Australia. Figures 3.2 shows the typical distributions of the project teams.
3.6.3 Interviews and Observations

Qualitative research studies often adopt interview as the main data collection technique in order to gather empirical data from the participants [57, 93]. Though Grounded Theory works with any data, whatever the source, whether interviews, observations or documents [106], interviewing is one of the most common and most powerful ways used to understand human social activity [93].

Interview questions can be open to capture a broad range of concerns of the participants, or closed to collect factual information with a specific set of alternative answers [255]. Interviews are divided into unstructured, semi-structured, or fully-structured [252]. In an unstructured interview, questions are formulated based on researcher’s interest on the phenomenon that is studied. In a fully structured interview, questions are planned and well established in advance, and all questions are asked in the same order as in the plan. In a semi-structured interview, the initially planned questions can evolve based on the exploration of research phenomena, and the researcher can decide which order the different questions are handled.
Kvale [178] defines a semi-structured interview as “an interview where the purpose is to obtain descriptions of the life world of interviewee with respect to interpreting the meaning of the described phenomena”.

Interviewing is a useful data collection technique for studies that focus on the meaning of particular phenomena to the participants [93, 252]. Semi-structured interviews with open-ended questions thus fit this research study because we needed to capture a broad range of concerns of distributed teams in Agile software development and to allow the interview questions to evolve based on the emerging theory. We often pre-arranged interviews with the participants. The interview was scheduled for an hour at a mutually agreed date, time and venue. We conducted face-to-face, one-on-one, semi-structured interviews with participants using open-ended questions. The English language was not the native language for the majority of the participants. Due to the fact that English was the lingua franca for them to communicate with one another in the distributed teams, these participants were quite well-versed in English. Hence, we used English for interviewing every participant. Interviews were voice recorded with the consent from the participants so that we could concentrate on the conversation during the interviews rather than trying to write down everything as it is being said. Artinian, Giske, and Cone [16] explain that even seasoned researchers usually need to record interviews for listening again to make sure all the nuances are understood.

Initially, an interview guide with a set of questions was prepared to support a smooth discussion with the participants (see Appendix B). The questions mainly focused on the background of the projects, roles and responsibilities of the participants, challenges faced by participants and fellow members of the distributed teams, and strategies adopted to overcome the challenges:
• “Please describe the project – its domain, duration, Agile methods used, team size and distribution.”

• “Please describe your role and key responsibilities in the project.”

• “Did you face any challenges in this project?”

• “What did you do to overcome the challenges?”

Several key concerns emerged from the initial analysis: culture, communication, trust, knowledge and senior management support. Using theoretical sampling, the interview questions gradually evolved to focus on these emergent concerns. In subsequent interviews, the following questions were also included to the interview guide:

• “How did you bridge cultural differences between team members from different countries and cultures?”

• “How did you communicate with team members across different sites?”

• “How did you collaborate with team members across different sites?”

• “How did you build trust with team members across different sites?”

• “How did you manage the knowledge required for a project?”

Besides interviewing, field observation is also an effective data collection technique for qualitative research [93, 252]. During field observations, researchers record information about human behaviour within a context using data from the senses [217]. Observations provide insights into research phenomena through observing the interaction between individuals and understanding the ongoing dynamic activities within the context [99]. Observation has the advantage of allowing researchers to see what happens at first hand, rather than just relying on what people say they do.
Observation is used in research in two ways – *structured* and *unstructured* – depending on the paradigm underlying the research [217, 234]. In structured observations, researchers observe events and behaviours using a guide that has been planned in advance. Researchers using a structured observation technique often use a permanent or semi-permanent record of an observation session on film, videotape, digital recording [99]. In unstructured observations, however, a researcher does not search for predetermined behaviours but rather enters the field with no predetermined notions as to the discrete behaviours that might be observed [217]. Unstructured observations are often less objective than structured observation because the researchers may be focusing on behaviours without a clear theoretical framework, and may not be coding the behaviours in a reliable, that is, repeatable way [99]. Positivist research generally uses structured observation, whereas interpretivist research uses unstructured observation [217].

In this research study, we conducted unstructured observations of seven different teams – five in India, one in the USA, and one in Australia. We particularly observed the set-up of the workplace such as video conferencing rooms, the Scrum artifacts such as burndown charts, the Scrum ceremonies such as daily stand-ups, and the XP practices such as pair programming. Figures 3.3 and 3.4 shows the Scrum artifacts from a project.

We used unstructured observations mainly to check whether “*what people say they do is the same as what they actually do*”. From the observations, we were able to validate the authenticity of the interview data. These observations supplemented the interviews, and provided a greater understanding of the practices that participants described during interviews. Field notes written during the observations were treated as another source of data that need to be analysed together with the interview data.
3.6.4 Research Context

This research is based on interviews with 55 participants from 38 different software companies in the USA, India and Australia, and field observations of seven Agile teams. Due to privacy and ethical consideration, we will only identify the participants using the codes P1 to P55. Table 3.1 shows the participant and project details.

Participants come from different roles in Agile software development such as Developer, Scrum Master, Agile Coach, Business Analyst, Quality Analyst (Application Tester), Product Owner, and Senior Manager (e.g. Human Resource Manager, Vice President (Delivery), Director of Technology). A majority of the participants have held different roles in different Agile software development projects either in the current or previous organisation. In general, every participant has been working on Agile software development projects for at least four years.
Table 3.1: Summary of Participants, Teams, and Projects. (Agile Role: Scrum Master (SM), Agile Coach (AC), Developer (DEV), Business Analyst (BA), Quality Analyst (QA), Product Owner (PO), Senior Manager (MGT))

<table>
<thead>
<tr>
<th>Code</th>
<th>Location</th>
<th>Position</th>
<th>Distribution</th>
<th>Size</th>
<th>Domain</th>
<th>Agile Methods</th>
<th>Duration (months)</th>
<th>Spat (weeks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>USA</td>
<td>DEV</td>
<td>N.America-Asia</td>
<td>8 to 10</td>
<td>Financial Services</td>
<td>Scrum</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>USA</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>12 to 14</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>P3</td>
<td>USA</td>
<td>SM</td>
<td>N.America-Europe-Asia</td>
<td>10</td>
<td>Mobile Application</td>
<td>Scrum</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>P4</td>
<td>USA</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>10</td>
<td>Online Trading</td>
<td>Scrum &amp; XP</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>P5</td>
<td>USA</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>8</td>
<td>Internet Media</td>
<td>Scrum &amp; XP</td>
<td>12</td>
<td>2 to 3</td>
</tr>
<tr>
<td>P6</td>
<td>USA</td>
<td>DEV</td>
<td>N.America-Europe</td>
<td>20 to 22</td>
<td>Internet Hosting</td>
<td>Scrum &amp; XP</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>P7</td>
<td>USA</td>
<td>AC</td>
<td>N.America-S.America-Asia</td>
<td>18</td>
<td>Internet Domain</td>
<td>Scrum &amp; XP</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>P8</td>
<td>USA</td>
<td>DEV</td>
<td>N.America-Asia</td>
<td>9 to 10</td>
<td>Publishing</td>
<td>Scrum &amp; XP</td>
<td>14</td>
<td>2 to 3</td>
</tr>
<tr>
<td>P9</td>
<td>USA</td>
<td>DEV</td>
<td>N.America-Europe</td>
<td>14</td>
<td>Web Search Engine</td>
<td>24</td>
<td>2 to 3</td>
<td></td>
</tr>
<tr>
<td>P10</td>
<td>USA</td>
<td>SM</td>
<td>N.America-S.America-Asia</td>
<td>10 to 12</td>
<td>Software Platform</td>
<td>Scrum</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>P11</td>
<td>India</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>13</td>
<td>Web Services</td>
<td>Scrum</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>P12</td>
<td>India</td>
<td>DEV</td>
<td>N.America-Asia</td>
<td>12</td>
<td>Internet Hosting</td>
<td>Scrum &amp; XP</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>P13</td>
<td>India</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>17 to 20</td>
<td>Web Portal</td>
<td>Scrum &amp; XP</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>P14</td>
<td>India</td>
<td>DEV</td>
<td>N.America-Asia</td>
<td>14 to 17</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>36</td>
<td>2</td>
</tr>
<tr>
<td>P15</td>
<td>India</td>
<td>QA</td>
<td>N.America-Asia</td>
<td>16</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>P16</td>
<td>India</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>16</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>P17</td>
<td>India</td>
<td>DEV</td>
<td>N.America-Asia</td>
<td>16</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>P18</td>
<td>India</td>
<td>BA</td>
<td>Europe-Asia</td>
<td>8</td>
<td>Financial Services</td>
<td>Scrum &amp; XP</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>P19</td>
<td>India</td>
<td>DEV</td>
<td>N.America-Asia</td>
<td>8 to 10</td>
<td>Insurance</td>
<td>Scrum</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>P20</td>
<td>Australia</td>
<td>MGT</td>
<td>Oceania-Asia</td>
<td>9 to 12</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>12</td>
<td>2 to 3</td>
</tr>
<tr>
<td>P21</td>
<td>Australia</td>
<td>SM</td>
<td>N.America-Oceania</td>
<td>9 to 12</td>
<td>Financial Services</td>
<td>15</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>P22</td>
<td>Australia</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>7 to 8</td>
<td>Power Distribution</td>
<td>Scrum</td>
<td>12</td>
<td>2 to 3</td>
</tr>
<tr>
<td>P23</td>
<td>India</td>
<td>QA</td>
<td>N.America-Asia</td>
<td>9</td>
<td>Automobile</td>
<td>Scrum &amp; XP</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>P24</td>
<td>India</td>
<td>AC</td>
<td>Europe-Asia</td>
<td>9</td>
<td>Automobile</td>
<td>Scrum &amp; XP</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>P25</td>
<td>India</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>24</td>
<td>Information Security</td>
<td>Scrum &amp; XP</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>P26</td>
<td>India</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>16</td>
<td>Healthcare</td>
<td>Scrum &amp; XP</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P27</td>
<td>India</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>30</td>
<td>Financial Services</td>
<td>Scrum &amp; XP</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>P28</td>
<td>India</td>
<td>MGT</td>
<td>N.America-Asia</td>
<td>20</td>
<td>E-Commerce</td>
<td>Scrum</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>P29</td>
<td>India</td>
<td>MGT</td>
<td>N.America-Asia</td>
<td>14</td>
<td>Social Networking</td>
<td>Scrum &amp; XP</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>P30</td>
<td>India</td>
<td>AC</td>
<td>Europe-Asia</td>
<td>8 to 10</td>
<td>Retail</td>
<td>Scrum &amp; XP</td>
<td>5</td>
<td>2 to 3</td>
</tr>
<tr>
<td>P31</td>
<td>India</td>
<td>AC</td>
<td>Europe-Asia</td>
<td>15 to 20</td>
<td>Retail</td>
<td>Scrum &amp; XP</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>P32</td>
<td>India</td>
<td>MGT</td>
<td>Europe-Africa</td>
<td>12</td>
<td>Retail</td>
<td>Scrum &amp; XP</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>P33</td>
<td>India</td>
<td>AC</td>
<td>Oceania-Europe-Asia</td>
<td>50</td>
<td>Recruitment</td>
<td>Scrum &amp; XP</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>P34</td>
<td>India</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>6 to 8</td>
<td>Real Estate</td>
<td>Scrum &amp; XP</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>P35</td>
<td>India</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>8</td>
<td>Online Payment</td>
<td>Scrum &amp; XP</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>P36</td>
<td>India</td>
<td>QA</td>
<td>N.America-Asia</td>
<td>10 to 15</td>
<td>Web Services</td>
<td>18</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>P37</td>
<td>India</td>
<td>DEV</td>
<td>Europe-Asia</td>
<td>16</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>P38</td>
<td>India</td>
<td>BA</td>
<td>N.America-Asia</td>
<td>12</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>P39</td>
<td>India</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>22 to 25</td>
<td>Telecommunication</td>
<td>Scrum &amp; XP</td>
<td>6 to 7</td>
<td>2</td>
</tr>
<tr>
<td>P40</td>
<td>India</td>
<td>DEV</td>
<td>Oceania-Asia</td>
<td>7</td>
<td>Online Trading</td>
<td>Scrum</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>P41</td>
<td>India</td>
<td>AC</td>
<td>Europe-Asia</td>
<td>18 to 12</td>
<td>Retail</td>
<td>Scrum &amp; XP</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>P42</td>
<td>India</td>
<td>MGT</td>
<td>N.America-Asia</td>
<td>7</td>
<td>Healthcare</td>
<td>Scrum</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P43</td>
<td>India</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>7</td>
<td>E-Commerce</td>
<td>Scrum &amp; XP</td>
<td>2 to 3</td>
<td></td>
</tr>
<tr>
<td>P44</td>
<td>India</td>
<td>PO</td>
<td>N.America-Asia</td>
<td>10 to 12</td>
<td>Cloud Computing</td>
<td>Scrum</td>
<td>4</td>
<td>2 to 3</td>
</tr>
<tr>
<td>P45</td>
<td>India</td>
<td>MGT</td>
<td>N.America-Asia</td>
<td>10</td>
<td>Financial Services</td>
<td>Scrum</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>P46</td>
<td>India</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>16</td>
<td>IT Services</td>
<td>Scrum</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>P47</td>
<td>India</td>
<td>AC</td>
<td>N.America-Asia</td>
<td>20</td>
<td>Financial Services</td>
<td>Scrum</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>P48</td>
<td>India</td>
<td>MGT</td>
<td>N.America-Europe-Asia</td>
<td>12</td>
<td>Online Media</td>
<td>Scrum &amp; XP</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>P49</td>
<td>India</td>
<td>AC</td>
<td>N.America-Europe</td>
<td>16</td>
<td>Telecommunication</td>
<td>Scrum</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>P50</td>
<td>India</td>
<td>PO</td>
<td>Europe-Asia</td>
<td>8</td>
<td>Web Services</td>
<td>Scrum</td>
<td>30</td>
<td>2 to 3</td>
</tr>
<tr>
<td>P51</td>
<td>India</td>
<td>SM</td>
<td>N.America-Asia</td>
<td>18</td>
<td>Web Application</td>
<td>Scrum &amp; XP</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>P52</td>
<td>India</td>
<td>SM</td>
<td>N.America-Africa</td>
<td>18</td>
<td>Web Application</td>
<td>Scrum</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>P53</td>
<td>India</td>
<td>AC</td>
<td>USA-Europe-Asia</td>
<td>10</td>
<td>IT Services</td>
<td>Scrum</td>
<td>5 to 6</td>
<td>2</td>
</tr>
<tr>
<td>P54</td>
<td>India</td>
<td>DEV</td>
<td>N.America-Asia</td>
<td>8</td>
<td>Education Services</td>
<td>Scrum</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>P55</td>
<td>India</td>
<td>DEV</td>
<td>N.America-S.America-Asia</td>
<td>7 to 8</td>
<td>IT Services</td>
<td>Scrum</td>
<td>(12)</td>
<td>3</td>
</tr>
</tbody>
</table>
Participants were practicing mainly Scrum or a combination of Scrum and XP through adoption of fundamental Agile practices such as small releases, continuous integration, estimating user stories, iteration planning, daily standup, release planning, sprint reviews and sprint retrospectives. Several participants were notably active in communities of practice where they have been participating in open forums, writing technical blogs, and speaking at local events and international conferences.

The project domains include telecommunication, health, finance, retail, education, human resource, insurance, trading, consultancy, and online media. Iteration varied from two to four weeks, and project duration varied from 3 to 36 months (ongoing projects are written within brackets in Table 3.1). According to the participants, projects often started with a small team size, typically with 6 to 12 members, but teams often scaled up in order to accommodate the increasing complexity of the projects.
3.7 Data Analysis

In Grounded Theory, data analysis – called coding – represents interpretation, conceptualisation, and categorisation of data, and its reintegration to generate theory [114]. Glaser and Strauss [114] assert that coding should be performed with an open mind without preconceived ideas. Researchers should not force preconceived ideas on the data by looking for evidence to support established ideas.

The basis of data analysis in Grounded Theory is the concept-indicator model [103]. An indicator refers to a word, phrase, or sentence, or a series of words, phrases, or sentences in the data being analysed, whereas a concept is a label or name for a higher level of abstraction that is associated with an indicator or related indicators [103, 182]. As a result of coding, two types of codes are produced: substantive codes and theoretical codes. Substantive codes are the emergent categories and properties that conceptually describe the phenomenon being studied, whereas the theoretical codes are the emergent abstractions that model the integration of substantive codes as an interrelated set of hypothesis for resolving the main concern [103, 111]. Open coding and selective coding give rise to substantive codes, whereas theoretical coding gives rise to theoretical codes [103, 108].

Glaser and Strauss [114] originally did not name the data analysis process as open coding, selective coding, or theoretical coding, but rather emphasised the constant comparison method for generating theory in the substantive area being studied. The original Grounded Theory [114] describes, “the constant comparison method is designed to aid the analyst ... in generating a theory that is integrated, consistent, plausible, close to the data”. It was Glaser [103] in Theoretical Sensitivity who identified the process of comparing indicator to indicator to generate categories and comparing new indicators to these emergent categories as “substantive coding” (which in-
CHAPTER 3. RESEARCH DESIGN

includes both open and selective coding), and the process of conceptualising how the substantive codes may relate to each other as hypothesis to be integrated into a theory as “theoretical coding”. Schwandt [267] describes the process of data analysis in Grounded Theory, including the role of theoretical sampling, in theory development:

“Grounded Theory requires a concept-indicator model of analysis, which in turn employs the method of constant comparison. Empirical indicators from the data (actions and events observed, recorded, or described in documents in the words of interviewees and respondents) are compared, searching for similarities and differences. From this process, the analyst identifies underlying uniformities in the indicators and produces a coded category or concept. Concepts are compared with more empirical indicators and with each other to sharpen the definition of the concept and to define its properties. Theories are formed from proposing plausible relationships among concepts and sets of concepts. Tentative theories or theoretical propositions are further explored through additional instances of data. The testing of the emergent theory is guided by theoretical sampling. Theoretical sampling means that the sampling of additional incidents, events, activities, populations, and so on is directed by the evolving theoretical constructs. Comparisons between the explanatory adequacy of the theoretical constructs and these additional empirical indicators go on continuously until theoretical saturation is reached (i.e., additional analysis no longer contributes to anything new about a concept). In this way, the resulting theory is considered conceptually dense and grounded in the data” [267].

In the following sections, we describe in detail the coding processes – open coding and selective coding that leads to substantive codes, and theoretical coding that leads to theoretical codes.
3.7. DATA ANALYSIS

3.7.1 Open Coding

Glaser [108] describes “open coding is the initial step of theoretical analysis that pertains to the initial discovery of categories and their properties”. Open coding breaks down, examines, compares, conceptualises and categorises the data [101]. Allan [10] suggests the use of key point technique for open coding instead of word-by-word or line-by-line techniques which can be very time consuming and potentially lead to confusion.

Interview transcripts and observation notes were analysed using key point open coding technique to explore meaning in the data by searching for similarities and differences [10, 11]. This technique requires the researcher to select points that are key to addressing research questions or concerns of the participants [10]. An example of analysis from interview transcript to the emergent category Trust is presented to explain the key point open coding. To begin data analysis with key point open coding, a key point from the data was identified and assigned a code to each key point. A code is a summary phrase, often 2 or 3 words, that implicitly describes a key point in the data. Using key point coding, researchers can trace back, through the interview transcriptions, to the actual quote and context of each key point.

The example below illustrates data analysis of the interview transcript from participant P1. “K” indicates “key point”, and suffix “P1” identifies participant P1 (i.e. the identifier for key point 8 made by participant P1 is KP18):

**Interview quotation:** “With online chats, we know whose is available and we often chat with one another.”

**KP18:** Team members engage in casual chats.

**Code:** casual chats
We identified many key points from every interview data. For example, we identified 42 key points from the interview data of participant P1. Table 3.2 shows examples of key points and codes from interview data of participant P1.

### Table 3.2: Examples of Key Points and Codes - Data from Participant P1

<table>
<thead>
<tr>
<th>ID</th>
<th>Key Point</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>K\textsubscript{P1}\textsubscript{3}</td>
<td>Video conferencing captures the visual aspect in communication</td>
<td>communication technique</td>
</tr>
<tr>
<td>K\textsubscript{P1}\textsubscript{6}</td>
<td>Team members were not aware of different cultures</td>
<td>cultural differences</td>
</tr>
<tr>
<td>K\textsubscript{P1}\textsubscript{8}</td>
<td>Team members engage in casual chats</td>
<td>casual chats</td>
</tr>
<tr>
<td>K\textsubscript{P1}\textsubscript{12}</td>
<td>Some members were comfortable reporting to team representatives</td>
<td>team ambassador</td>
</tr>
<tr>
<td>K\textsubscript{P1}\textsubscript{19}</td>
<td>Team members didn’t want everyone to hear them asking questions</td>
<td>lack of openness</td>
</tr>
<tr>
<td>K\textsubscript{P1}\textsubscript{24}</td>
<td>Coach travelled to different locations to help team members</td>
<td>coach travels</td>
</tr>
<tr>
<td>K\textsubscript{P1}\textsubscript{35}</td>
<td>Team members were not engaged on the calls</td>
<td>lack of team spirit</td>
</tr>
</tbody>
</table>

Using the *constant comparison method* [107], each code was constantly compared with the codes from the same interview, and those from other interviews. The codes that were related to a common theme were grouped together to produce a higher level of abstraction called a *concept*. Using diagrams to illustrate the emergence of concepts and categories are recommended in Grounded Theory [10, 101]. Figure 3.5 shows emergence of the concept *Social Communication* from the codes.

![Figure 3.5: Example of emergence of a concept from codes](image-url)
3.7. DATA ANALYSIS

Many fresh concepts emerged as the codes were continuously compared to each other. These concepts were also analysed using constant comparison method, and the concepts that related to a common theme were grouped together to produce another higher level of abstraction called a category. Figure 3.6 shows emergence of the category Trust from underlying concepts.

![Diagram showing emergence of Trust category from underlying concepts]

Figure 3.6: Example of emergence of a category from underlying concepts

The continuation of data collection through theoretical sampling directly translates to an increasing amount of interview data that need to be analysed. Glaser [103, 109] suggests that researchers ask a set of questions of the data from the beginning of open coding: “what is the data study of?”, “what is actually happening in the data?”, “what is the main concern being faced by the participants?”, “what accounts for the continual resolving of this concern?”, and “what category does this incident indicate?”. These questions continually remind the researchers of the actual purpose of the research study, and keep them theoretically sensitive when collecting and analysing the data [103].
Figure 3.7 shows the levels of data abstractions [140, 201] in Grounded Theory. The analysis gave rise to many other codes, concepts and categories. Throughout this thesis, emergence of different concepts and categories is illustrated using diagrams in a similar manner. A full coding structure (showing the codes, concepts and categories) for the grounded theory that emerged from data analysis is provided in Appendix D.

![Diagram of levels of abstractions in Grounded Theory]

Figure 3.7: Levels of Abstractions in Grounded Theory

### 3.7.2 Core Category

The goal of Grounded Theory is to generate a relevant theory that accounts for a pattern of behaviour about what is going on in the area studied [18, 103]. In discovering a theory, the researcher continues open coding until the emergence of a core category. Glaser [103] describes that a core category “accounts for a large portion of the variation in a pattern of behaviour”.
3.7. DATA ANALYSIS

The core category does not necessarily account for all of the behaviour under study, but rather accounts for one particular behaviour that is highly relevant for participants in the substantive area [109]. The emergence of a core category reflects that the researcher has conceptually identified the most important concern which participants continually resolve [103]. In other words, the core category turns out to be the research problem for the phenomenon being studied within its context [103, 186]. Glaser [103] describes that “the generation of a theory occurs around a core category”. It is through the articulation and explanation of a core category and its properties that a dense and saturated substantive theory will emerge [103].

There are eleven criteria for establishing a category as the core [103]. For example, a core category must be central and connects with other categories quickly and richly, it takes more time to saturate as it reoccurs frequently in the data, and it must relate meaningfully and easily with other categories. The emergent category that passed all the criteria for core was “becoming one team”. This core is a processural type of category known as basic social process (BSP). Glaser [103] explains that “all BSPs are core categories, but not all core categories are BSPs” because while a core category can be used with any theoretical code, only BSPs “give the feeling of process”. Glaser [103] describes that “a process is an action which occurs over time and involves change over time”. A BSP explains the preponderance of behaviour in a substantive area of the research, occurs over time, place and people, and lends itself to change over time in the light of new data [106, 110]. With the discovery of a BSP as the core category for our research, we switched the focus from studying the social units (research participants) to studying the process (becoming one team). In this study, the generation of the substantive theory happened around the basic social process of becoming one team.
3.7.3 Selective Coding

Selective coding begins after the researcher has established a core category. The researcher ceases open coding, and selectively codes for the core category by delimiting data collection, analysis and theoretical sampling only to those categories that relate to the core category [103, 140]. At this point, coding appeared much easier because we were very familiar with the constant comparison method, and only the most pertinent key points of the interview transcripts are used and coded.

In this study, selective coding resulted in a set of categories that conceptualise and explicate the basic social process of becoming one team. Selective coding was continued until we have sufficiently elaborated and integrated the core category, its properties, and its theoretical connections to other relevant categories [103].

3.7.4 Memoing

Glaser and Strauss [114] emphasised the need to write memos of theoretical ideas that occur during the data analysis process in order to move the coding away from the descriptive and towards the theoretical. A theoretical memo is defined as “the theorising write-up of ideas about substantive codes and their theoretically coded relationships as they emerge during coding, collecting and analyzing data” [109]. Glaser [109] explains that memoing, or writing theoretical memos, is the “core stage” of a Grounded Theory study, and a researcher who skips this stage is not doing a Grounded Theory. Glaser [103] suggests that researchers should “always interrupt coding or data recording for writing a memo, when an idea occurs, so the idea is not lost”.

The four basic goals in memoing are to theoretically develop ideas, with complete freedom into a memo fund that is highly sortable [103]:
3.7. DATA ANALYSIS

**Ideas:** Memoing provides an avenue for the researcher to express ideas and thoughts about an emergent code, concept, or category in order to raise the data to a conceptual abstraction. This ideation presents the connections between emergent categories and their properties, and integrates them to generate the theory [103].

**Freedom:** Memoing does not accord importance to good prose, sentence construction, punctuation, or even the language that researchers use to record the ideas. The only thing that matters is to “get the ideas out” [103]. This freedom in memoing allows researchers to work faster by focusing on the idea only and not its presentation.

**Memo fund:** Memoing should be proliferated on all emergent categories in order to build a large collection of memos that can be used to generate a dense rich theory. This “timeless” memo fund remains open so that researchers can return to the fund and re-work with the theoretical ideas and connections [103].

**Highly sortable:** Memos have to sort very quickly according to the theoretical ideas of the emergent categories. That is, the memos are written in a manner that promotes sortibility. For example, each memo has a title or caption, categories that appear in a memo are highlighted, or writing the memos on index cards [103].

In this study, we continuously wrote-up memos as they emerge throughout data analysis. We divided the memos into two groups: (i) *concept memos*, which describe the codes, concepts, categories and the connections between them, and (ii) *process memos*, which describe the activities and decisions leading to the emerging theory. Whenever an idea about a code, concept, category or process emerges, we stop coding and begin writing a memo which represents the ideas developing from the data. We created
separate files to record memos on different topics so that we can easily retrieve and edit them when needed. We provided each memo with a title and highlighted the concepts that appeared in the memo. Figure 3.8 and Figure 3.9 are examples of concept memos, whereas Figure 3.10 and Figure 3.11 are examples of process memos.

Figure 3.8: Concept Memo on Team Interaction

Besides the data from interviews and observations, we also used data from other sources such as literature from substantive area, documents, discussion with research supervisors, mentors and fellow researchers, and even personal experience to write theoretical memos. We revisited the memos when we need to modify them as we grasp better understanding and realisation on a particular topic. We also kept reminding ourselves that the main focus of the memos was the conceptual connections between the emergent categories.
3.7. **DATA ANALYSIS**

**Planned Unconference:**

This is a casual meeting, purposefully planned to facilitate interaction between team members. It provides team members with an opportunity to have a casual conversation, which is important to build rapport and foster solid relationships with one another. With planned unconference, distributed teams emulate co-located teams, in which team members engage in casual conversations on a regular basis. Some teams allocate a short duration of time (i) before daily meetings (P1,P4,P24,P53), (ii) before retrospectives (P23,P42,P46), and (iii) before the knowledge sharing sessions (P28,P34,P38).

![Figure 3.9: Concept Memo on Planned Unconference](image)

**Participant Position: Senior Manager**

Based on interviews with 18 participants so far (10 from the USA and 8 from India), we found that integrating Agile methods in global software development requires considerable effort from senior management in the organisations. We have not interviewed senior managers yet. We must find ways to interview senior managers from the USA and India in order to understand the key responsibilities of senior management for the success of Agile teams. We can triangulate the data (with regard to the support from senior management) from different sources (participant position).

![Figure 3.10: Process Memo on Participant Position: Senior Manager](image)
3.7.5 Sorting

Once theoretical saturation was reached, the researcher can begin sorting the theoretical memos according to the conceptual ideation of the memos. *Theoretical sorting* of memos is an “essential step” for formulating the theory for presentation or writing because “it begins to put the fractured data back together” [103]. Glaser [103] further explains that researchers should conceptually sort the ideas and *not* the data. Theoretical sorting of ideas forces the researcher to remain on a conceptual level and think in terms of theoretical coding as categories are integrated into a theory [103]. Glaser [103] suggests several *analytical rules* related to starting to sort, the core category, fit and conceptual level, completeness, mechanics of sorting, and theoretical pacing to guide the theoretical sorting and the subsequent writing of the theory as it emerges. When all the memos have been sorted, *theoretical completeness* is reached. Theoretical completeness implies theoretical coverage where the researcher has explained the research phenomenon with
“the fewest possible concepts, and with the greatest possible scope, as much variation as possible in the behaviour” [103]. The conceptually sorted memos richly explicate the research phenomenon and exhibit strong connections between the categories and their properties [103, 108].

In order to begin theoretical sorting of memos, we printed out the memos for each topic from the separate files and glued each memo on an index card, creating a stack of ‘memo cards’. We sorted the memos so that the topics relate to one another in a relevant order that can be used to generate a theory. We began sorting all other categories and properties only as they relate to the basic social process of “becoming one team”. We sorted on two conceptual levels: on the first level, the memos with the same conceptual titles were grouped together, generating an outline of the theory, and on the second level, memos in each group were sorted so that all ideas fit within the groups in the outline. This theoretical outline was used to write up the theory. Figures 3.12 shows theoretical sorting of memos.
3.7.6 Theoretical Coding

The final step of data analysis is the theoretical coding that systematically generates the theoretical codes. Theoretical coding is the process of “coding and constant comparative analysis that yields the conceptual relationship between categories and their properties as they emerge” [108]. Theoretical coding integrates all the data, codes, concepts and categories into a set of seamless theoretical codes. Substantive codes conceptualise the empirical substance of the area of research, whereas theoretical codes implicitly conceptualise how the substantive codes will relate to each other as a set of interrelated hypotheses to be integrated into the theory [103]. Although theoretical codes are not strictly necessary to develop a substantive theory, Glaser argues that without theoretical codes the emergent theory is usually “confused, unclear theoretically, and/or typically connected by descriptive topics but going nowhere theoretically” [111].

A Grounded Theory is best when theoretical codes are used because they effectively weave the ‘fractured’ substantive codes into an organised theory [103]. Choosing the appropriate way to link categories in a Glaserian GT depends on what emerges during coding, and the way that the core category appears to relate to the other categories around it. Glaser [108] suggests researchers systematically code for categories and properties, and let suitable theoretical codes emerge from the analysis.

In this study, several substantive codes emerged around the basic social process of becoming one team. Using theoretical coding, we analysed the conceptual relationship between these emergent categories. The substantive codes Same Team, Same Time, Same Space, Same Practices, and Same Culture gave rise to the theoretical code “strategies”. Figures 3.13 shows the emergence of the theoretical code “strategies” from the underlying substantive codes. The arrow illustrates the substantive code gave rise to the theoretical code.
Similarly, the substantive codes *Trust* and *Senior Management Support* gave rise to the theoretical code “impact factors”. Figures 3.14 shows the emergence of the theoretical code “impact factors” from the underlying substantive codes. The arrow illustrates the substantive code gave rise to the theoretical code.
Glaser [103, 109, 111] discusses many ways that substantive and theoretical codes can be integrated and organised into a grounded theory. Glaser [103, 109, 111] suggests almost 50 different theoretical coding families such as the Strategy family, Process family, Degree family, Dimension family and Type family for assisting researchers to conceptualise how the categories and their properties may relate to each other. Figures 3.15 shows examples of theoretical coding families with detailed descriptions [103].

By comparing the emergent theoretical codes with the theoretical coding families, the Strategy coding family emerged to be the best ‘fit’ for the basic social process of becoming one team and its properties (s 10.5). The Strategy is applied when there is a specific strategy that can be adopted to resolve a problem in the research phenomenon [111]. This coding family also sharpened the relationships between substantive codes and provided a vocabulary for describing these relationships as we began to write-up the emergent theory.

In this study, each substantive code which forms the theoretical code “strategies” describes a strategy to bridge either the temporal, spatial or socio-cultural distances in distributed software development, whereas each substantive code which forms the theoretical code “impact factors” describe a factor which influences the adoption of the strategy for a distributed team. We provide a detailed description of these strategies and factors in chapter 4.

### 3.8 Generating a Theory

A grounded theory is presented “either as a well-codified set of propositions, or in a running theoretical discussion, using conceptual categories and their properties” [114]. There are basically two kinds of grounded theory – substantive and formal – where both theories are grounded in data.
<table>
<thead>
<tr>
<th>Families</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Six C’s</td>
<td>Causes (sources, reasons, explanations, accounting or anticipated consequences), Context or Ambiance, Contingencies, Consequences (outcomes, efforts, functions, predictions, anticipated/unanticipated), Covariances, Conditions or Qualifiers.</td>
</tr>
<tr>
<td>Process</td>
<td>Stage, Staging, Phases, Phasing, Progressions, Passages, Gradation, Transitions, Steps, Ranks, Careers, Ordering, Trajectories, Chains, Sequencing, Temporalizing, Shaping, Cycling.</td>
</tr>
<tr>
<td>Degree</td>
<td>Limit, Range, Intensity, Extent, Amount, Polarity, Extreme, Boundary, Rank, Grades, Continuum, Probability, Possibility, Level, Cutting Points, Critical Juncture, Statistical Average (mean, medium, mode), Deviation, Exemplar, Modicum, Full, Partial, Almost, Half.</td>
</tr>
<tr>
<td>Dimension</td>
<td>Dimensions, Elements, Divisions, Piece of, Properties of, Facet, Slice, Sector, Portion, Segment, Part, Aspect, Section.</td>
</tr>
<tr>
<td>Type</td>
<td>Type, Form, Kinds, Styles, Classes, Genre.</td>
</tr>
<tr>
<td>Theoretical</td>
<td>Parsimony, Scope, Integration, Density, Conceptual level, Relationship to data, Relationship to other theory, Clarity, Fit, Relevance, Modifiability, Utility, Condensibility, Inductive-Deductive balance and Interfeeding, Degree of, Multivariate structure, Use of theoretical codes, Interpretive, Explanatory, Predictive Power.</td>
</tr>
<tr>
<td>Mainline</td>
<td>Social control, Recruitment, Socialization, Stratification, Status passage, Social organization, Social order, Social interaction, Social mobility.</td>
</tr>
<tr>
<td>Interactive</td>
<td>Mutual Effects, Reciprocity, Mutual Trajectory, Mutual Dependency, Interdependence, Interaction of Effects, Covariance, Face to Face Interactions, Self-Indications, Delayed-Interaction.</td>
</tr>
<tr>
<td>Means-goal</td>
<td>End, Purpose, Goal, Anticipated consequences, Products.</td>
</tr>
<tr>
<td>Cultural</td>
<td>Social norms, Social values, Social belief, Social Sentiments.</td>
</tr>
<tr>
<td>Cutting Point</td>
<td>Boundary, Critical juncture, Cutting point, Turning point, Benchmark, Division, Cleavage, Scales, in-out, intra-extra, Tolerance levels, Dichotomy, Trichotomy, Polychotomy, Deviance, Point of no return.</td>
</tr>
<tr>
<td>Consensus</td>
<td>Clusters, Agreements, Contracts, Definitions of Situation, Uniformities, Opinions, Conflict, Discensus, Differential perception, Cooperation, Homogeneity-heterogeneity, Conformity, Non conformity.</td>
</tr>
</tbody>
</table>

Figure 3.15: Examples of theoretical coding families.
A substantive theory is developed for a substantive, or empirical, area of research, whereas a formal theory is developed for a formal, or conceptual, area of research [114]. The distinction between the two theories lies in the different levels of generality, indicated by the breadth of focus – formal theories are less specific than substantive theories, and formal theories are developed from many substantive theories [114]. In this study, we developed a substantive theory which describes how a distributed teams in Agile software development becomes one team. Figures 3.16 shows the progression of theory development in Grounded Theory (adapted from [186]).

![Diagram of theory development process]

Figure 3.16: The progression of theory development in Grounded Theory (adapted from [186]).

There is no universal definition of the concept of an empirically-based theory nor is there a definitive answer to the question of what constitutes a theory [278, 308]. Glaser [108, 109] suggests a theory is a conceptual explanation for the pattern of behaviour of the participants seen in the substantive area. Weber [308] suggests a theory is a particular kind of model that is intended to account for some subset of phenomena in the real world.
3.8. GENERATING A THEORY

Denscombe [63] suggests a theory is a proposition about the relationship between constructs. Thomas and James [298] suggest a theory is a set of statements informing the world something new about the research phenomena and which can be proved or disproved by empirical investigation. Pentland [235] suggests a theory is an explanation of what is causing the observed outcomes.

In sciences that are relevant to empirical software engineering, discussions concerning theory tend to revolve around the following issues: (1) what a theory does, (2) what the elements of a theory are, (3) how theories are formed, and (4) how theories are evaluated [278]. Gregor [117] proposes a taxonomy that classifies information systems theories that includes the following categories – theory for analysing, theory for explaining, theory for predicting, theory for explaining and predicting, and theory for design and action:

*Theory for analysing* says “what is”. The theory does not extend beyond analysis and description, and no causal relationships among phenomena are specified and no predictions are made.

*Theory for explanation* says “what is, how, why, when, and where”. The theory provides explanations but does not aim to predict with any precision. There are no testable propositions.

*Theory for predicting* says “what is and what will be”. The theory provides predictions and has testable propositions but does not have well-developed justificatory causal explanations.
Theory for explaining and predicting says “what is, how, why, when, where, and what will be”. The theory provides predictions and has both testable propositions and causal explanations.

Theory for design and action says “how to do something”. The theory gives explicit prescriptions such as methods, techniques, principles of form and function for constructing an artifact.

Urquhart, Lehmann and Myers [302] suggest grounded theory has the capability to generate theory that exists in all these categories because it contains the essential building blocks of any theory: constructs in the form of categories and relationships between those constructs in the form of theoretical coding. In this study, we aimed to generate a theory for explaining the key concern of distributed teams in Agile software development. This theory provides an insight into the most important concern which our research participants continually resolve and brings about an understanding of how things are or why they are as they are [117].

Sjoberg et. al [278] argues software engineering research generally accepts that constructs and relationships between constructs constitute the basic building blocks of theories, and that it is important to delineate a theory’s area of application by specifying scope conditions. Whetten [309] suggests a theory must address four components: (1) what are the concepts, factors or constructs should be included in a theory; (2) how are the concepts related; (3) why are the concepts important to the phenomena; and (4) who, where, when are the temporal and contextual factors that delimits the theory. Hence, Whetten’s framework for generating a theory incorporates all salient aspects of a theory.
In this study, we adopted Whetten’s framework for generating a theory in order to generate a theory for explaining the research phenomenon. We developed a substantive theory that answers “what, how, and why” questions about the research phenomenon within its who, where, when context. The substantive theory, from here forth called “The Theory of One Team”, is fundamentally a conceptual explication of the basic social process of “becoming one team” based on the codes, concepts and categories that emerged from the data, and their inter-relationships with one another, within the context being studied. In the following chapters 4–8, we provide a detailed description of “The Theory of One Team”.

3.9 Evaluating the Theory

There is much dialogue centred on the difficulty of establishing validity criteria in qualitative research [14, 20, 277, 282, 310]. In this section, we describe possible approaches to validate our research study.

Lincoln and Guba [189] suggest criteria of trustworthiness for evaluating qualitative research studies. According to Lincoln and Guba [189], the criteria of trustworthiness has four aspects, namely credibility, dependability/auditability, confirmability, transferability:

**Credibility** refers to how much the data collected accurately reflects the realities of the research phenomenon [34, 189, 219, 275]. Lincoln and Guba [189] argue credibility is the most important criteria in establishing trustworthiness. Credibility is achieved through validity procedures such as triangulation of data, peer debriefing and member checking [6, 189, 275]. Triangulation may involve the use of a wide range of informants, different data collection techniques such as interviews, observation and focus groups, and different sites through participation of informants across different organisations. The main purpose of triangulation of data
is to get a comprehensive, stable view of ‘reality’ based on data from a wide perspective [190, 275]. A peer debriefing is the review of the data and processes within the study by an individuals who is familiar with the research method or the phenomenon being studied [58, 190, 275]. In member checking, data and the corresponding interpretations are given back to the participants in the study so that they can confirm the credibility of the information and narrative account. Lincoln and Guba [189] highlight member checking is the most crucial validity procedure for establishing credibility in a study.

**Dependability/Auditability** ensures data represents the changing conditions of the phenomenon under study [34, 189, 219, 275]. In order to address the dependability/auditability criteria objectively, Lincoln and Guba [189] suggest processes within the study should be reported in detail, such as the research design which describes what was planned and how the plan was executed, and the operational detail of data collection which addresses the minutiae of what was done in the field.

**Confirmability** examines whether research findings reflect informants perspective and conditions of the study rather than the researcher perspective [34, 189, 219, 275]. Lincoln and Guba [189] suggest processes within the study should ensure as far as possible that the findings reflect the experiences and ideas of the informants, rather than the preferences of the researcher.

**Transferability** is concerned with the applicability of the findings of one study to other contexts [34, 189, 219, 275]. Lincoln and Guba [189] suggest researchers provide sufficient contextual information within the research report of the context in which the work was undertaken, and it is the responsibility of the readers to determine how far they can be confident in transferring to other situations the results and conclusions presented.
Similarly, Creswell [57] outlined a number of “specific criteria” such as triangulation, member checking, and peer debriefing that must be employed in any study for validating qualitative research. While Lincoln and Guba [189] argue all four criteria must be attained for the research to be considered trustworthy, Creswell [57] suggest at least two criteria must be employed to validate a research study. Interpretive researchers, however, rule out methodology as the basis for validity because establishing such criteria continues the positivist assumption of an underlying objective reality to which research results can be compared and judged for truth [14, 222, 259, 282]. Nielsen [222] argues the adoption of positivist validation criteria to interpretive research is “in danger of making qualitative research into bad quantitative research”.

Reaching the desired goal and meeting the requirement of trustworthiness become particularly problematic for interpretive, qualitative researchers due to the considerable debate about what it means to do valid research in the field of qualitative research [14, 222]. Angen [14] suggests interpretive, qualitative research should consider validation from the ethical and substantive perspectives:

**Ethical validation:** Interpretive research depends on the inter-subjective creation of meaning and understanding, and the researcher is not separate from the researched. Therefore, the researcher’s values are inherent throughout the research process, and moral soundness becomes the basis for judging an interpretive research [14, 57]. Angen [14] argues validation is an ethical question that should be addressed from inception to completion of research. That is, validation should be considered when research is formulated, carried out, and written up.
Substantive validation: The substance of the research is an important focus for evaluating interpretive research [14]. Self-reflection contributes significantly to the substantive validation of the research results [58]. The researcher self-disclose personal assumption, beliefs, and biases, that shape the research, right from the beginning of the research. Interpretive research is a “chain of interpretations” that should be documented and produced to others as evidence of how the conclusions were reached in order to evaluate the trustworthiness of the meanings of the results [222].

Glaser [103, 108] argues a grounded theory does not require further testing because it comes directly from the data itself. Glaser [103, 108], however, suggests credibility of a grounded theory can be evaluated through four criteria, namely fit, work, relevance and modifiability:

Fit refers to “the ability of the categories and their properties to fit the realities under study in the eyes of the subjects, practitioners, and researchers in the area” [109]. That is, the categories and their properties, which define the theory, have naturally emerged from the data, and were not forced by preconceived ideas or hypothesis.

Work refers to “the ability of the theory to explain the major variations in behaviour in the area with respect to the processing of the main concerns of the subjects” [109]. That is, a grounded theory should be able to explain what happens in the data, predict what will happen, and interpret what is happening in the area studied.

Relevance is achieved when a grounded theory “allows core problems and processes to emerge” [103]. That is, participants and practitioners must consider that the theory is useful, meaningful and applicable to the people working in the substantive area.
3.9. **EVALUATING THE THEORY**

**Modifiability** refers to “the quality of the theory to be ready for changes to include variations in emergent properties and categories caused by new data” [109]. That is, a theory can continue to be extended by other researchers when more data and new ideas are compared to the existing theory. Modifiability is an “ever-ongoing process” [103], and all grounded theories have potential to evolve further.

*Fit* is considered the most important criterion for evaluating the credibility of a theory, whereas the criteria *work*, *relevance*, and *modifiability* supports the “fitness” of the theory and useful in the broader evaluation of the quality of the theory [194]. Hence, when considering credibility of a grounded theory, *fit* deserves particular attention:

“*Fit can either be interpreted from a realist position as a matter of correspondence to reality or it can be interpreted from one of the various nonrealist positions as a matter of coherence, consensus or pragmatic usefulness, depending on the theory of truth inherent to the specific nonrealist position*” [194].

Urquhart et al [302] argue theoretical sampling is the single most important contributor to the “fit” of a theory because theoretical sampling ensures the comprehensive nature of the theory – well grounded in data and explains “what is actually going on”. Without theoretical sampling, it will be impossible to establish how saturated the theory is.

Weber [308] proposes a framework and criteria that can be used to evaluate the quality of theories within the information systems discipline in order to inform the development of high-quality theories. Weber [308] argues a theory should be evaluated from two perspectives: (1) “parts” – the evaluation should focus on the quality of the individual components that make up the theory; (2) “whole” – the evaluation should focus on the quality of the theory considered in toto. Constructs, associations, states, and events constitute the *parts*: 
**Constructs** refers to attributes in general of some class of things in its domain. Weber [308] argues the classes of things to which attributes in general pertain ought to be defined precisely to ensure that the meanings of each class and the things in each class are clear. Further, the nature of each attribute that pertains to a particular class ought to be defined precisely.

**Associations** refers to relationships between constructs, which reflects a pattern that is hypothesised to hold across instances of things in the class or classes of things that the theory covers. Weber [308] suggests that when evaluating the meaning to ascribe to an association, it is important to reflect upon whether a theory covers only static phenomena, dynamic phenomena, or a combination of both static and dynamic phenomena. Further, it is also important to understand whether the constructs represent attributes in general of a single class of things or multiple classes of things.

**States** refers to a set of values for things that fall within a theory’s domain and for which a theory is intended to have explanatory and predictive power. Weber [308] argues a theory should specify clearly and as precisely as possible the state space of things in the class or classes of things that it is intended to cover.

**Events** refers to happenings that change over time in dynamic phenomena. Weber [308] argues all conceptually possible pairs of inside-boundary states must be considered in order to determine the conceivable event space covered by the theory.

A theory has emergent attributes – attributes of the theory as a whole rather than attributes of its parts. Importance, novelty, parsimony, level, and falsifiability constitute the **whole**:
3.9. EVALUATING THE THEORY

**Importance** of a theory, which is assessed via judgments made about the importance of its focal phenomena, which might be deemed important from the viewpoint of practice or research. Weber [308] argues there is little point to having a theory with rigorously specified constructs, associations, states, and events if it addresses uninteresting phenomena.

**Novelty** of the theory is an important factor determining the value ascribed to it by researchers, and the likelihood that papers describing the theory will be accepted for publication. Weber [308] argues judgments about a theory’s novelty or originality and judgments about its contributions to knowledge appear to be closely related.

**Parsimonious** theory achieve good levels of predictive and/or explanatory power in relation to their focal phenomena using a small number of constructs and associations. Weber [308] argues it is often easier to articulate the nature of states and events that fall within the boundary of the theory using a small number of constructs and associations.

**Level** describe the coverage of research phenomena, either narrow or broad. Weber [308] argues theories that cover a narrow, constrained set of phenomena, called “micro-level” theories, run the risk that they will be deemed uninteresting or unimportant. In contrast, theories that cover a broad range of phenomena, called “macro-level” theories, run the risk that constructs and associations are defined imprecisely.

**Falsifiability** of a theory requires the researchers to generate precise predictions about the focal phenomena in order to undertake empirical tests of the theory. Weber [308] argues not all theories can be proven via empirical tests because it is impossible to test the theory for all things, all instances of associations, all states, and all events that fall within its boundary.
In this study, we primarily adopted Angen’s [14] validation approaches for interpretive, qualitative research, and then creatively combined the researcher’s theoretical sensitivity (s 3.5.2) with Grounded Theory’s rigorous analysis process (s 3.7) to produce valuable interpretations. We also primarily adopted Glaser’s [103, 108] criteria for evaluating the credibility of a grounded theory. In section 10.7, these validation and credibility criteria are revisited to demonstrate how well our grounded theory evaluates against them. We also discuss how well our grounded theory evaluates against Lincoln and Guba’s [189] criteria of trustworthiness and Weber’s [308] criteria for evaluating the quality of a theory in order to strengthen and deepen our analysis of the validity of our qualitative research study.

3.10 Discussion

In this chapter, we outlined the research design for this study, which integrates the research paradigm, the research approach, and research method. In particular, we described in detail the Grounded Theory research method and its application in our study, and described several approaches to validate and evaluate the emergent theory.

One unique characteristic of Grounded Theory is that Grounded Theory can adopt any epistemological stance appropriate to the data and the ontological stance of the researcher in a particular research study. From inception to conclusion, Grounded Theory focuses towards an area of social concern, and uses everything in the research scene as data, whatever the source, the type, and the nature of data. We carried out this research study through the lens of the interpretive paradigm for the purpose of understanding the main concern of distributed teams in Agile software development from the perspective of the participants, and generating an
3.10. DISCUSSION

explication of the research phenomenon within its context. Throughout the research study, there was a natural synergy between Grounded Theory and Agile software development. For example, we found that Grounded Theory is suitable to be used in areas that are under-explored, such as Agile software development with distributed teams. We also found that Grounded Theory is ideal to study social interactions and behaviour of people in a given substantive area, such as Agile team.

The key to using Grounded Theory successfully is “trusting in emergence”. Ideally, researchers should trust that the research question will emerge in the initial stages of data analysis, and the main concern of participants will be discovered with the emergence of the core category. The researcher should also be prepared to leave professional interests aside in the interests of participants, and optimise the strength of constant comparison method to discover the pattern of behaviour in a particular research area. We found that reading literature from well-respected Grounded Theory research from different disciplines developed our trust in emergence and increased our understanding on the application of Grounded Theory. We also found that frequent discussions with mentors, research supervisors, fellow researchers and participants significantly contributed to stay focused on the data and discovery through emergence.

In the chapters 4–8, we provide a detailed description of “The Theory of One Team” and discuss the the research findings in the light of existing literature.
Chapter 4

The Theory of One Team

In chapters 4–8, we describe the theory and discuss the findings in the light of related literature. “The Theory of One Team” explains how a distributed team in Agile software development adopts explicit strategies for bridging spatial, temporal, and socio-cultural distances, while facing critical impact factors, in order to become one team. The strategies are Same Team, Same Time, Same Space, Same Practices and Same Culture, whereas the impact factors are Trust and Senior Management Support. The Same Team strategy aims to develop cohesion among members of a distributed team; the Same Time strategy aims to promote synchronous communication in order to bridge temporal distance; the Same Space strategy aims to promote temporary physical co-location in order to bridge spatial distance; the Same Culture aims to promote cross-cultural communication; the Same Practices strategy aims to promote the use of common practices – the Same Team, Same Culture and Same Practices strategies bridge socio-cultural distance. A distributed team in Agile software development adopts a particular strategy or a combination of strategies for bridging spatial, temporal, and socio-cultural distances in order to work together as one team, while trust and senior management support may influence the adoption of these strategies within the team. Figure 4.1 shows the high level abstraction of “The Theory of One Team”.

107
In this study, a *strategy* refers to “a high-level plan of action designed to achieve a particular goal”; a *tactic* refers to “a specific action designed to execute a strategy”; and an *impact factor* refers to “a circumstance that influence the adoption of these strategies”. Every strategy comprises a number of explicit tactics used to execute the strategy. Figure 4.2 shows the plan and goal of the strategies, whereas Figure 4.4, Figure 5.1, Figure 6.1, Figure 7.1 and Figure 7.2 show the tactics for the Same Team, Same Time, Same Space, Same Culture and Same Practices strategies respectively.
In general, many participants (P3, P5, P7, P16, P20–P24, P26, P27, P30, P40, P44, P45, P48, P53) explicitly used terms such as “one team”, “single team”, and “whole team” to point out that members of a distributed team strive to work together despite the spatial, temporal and socio-cultural distances in order to become one team:

“We have absolutely one single team [but] we are working in a different time zone. And, the client is definitely part of the team.” — P16, Scrum Master.

“When we roll out Agile projects, our focus is learning about team work. It’s always about how to build a single team that can work well together when the team is distributed.” — P44, Product Owner.
“The one team concept is really important. If you can achieve it, you can be really successful [because] people are going to be focused and working hard when they are working together as one team.” —P3, Scrum Master.

A large number of participants report distributed teams must get to know how to work together in software development projects:

“...a distributed team has to know how to work together with team members across different locations.” —P3, Scrum Master.

“...we explore different ways to work together although we are [located] at different sites.” —P40, Developer.


In the chapters 4–8, we describe how a distributed team in Agile software development become one team. Figure 4.3 shows “The Theory of One Team” in detail (with concepts that gave rise to the substantive codes – Same Team, Same Time, Same Space, Same Culture, Same Practices, Trust, Senior Management Support – and the association between the theoretical codes – Strategies, Impact Factors). We describe the Same Team strategy in this chapter; the Same Time strategy in chapter 5; the Same Space strategy in chapter 6; and the Same Practices and the Same Culture strategies in chapter 7. In chapter 8, we discuss how the impact factors influence the adoption on these strategies.
Figure 4.3: The Theory of One Team in detail.
4.1 Same Team

Same Team strategy aims to develop team cohesion among members of a distributed team.

Co-located teams worked effectively together because members of a co-located team knew one another from the professional and personal backgrounds, and thus, they developed solid relationships with one another. In contrast, when team members are dispersed across different geographical locations, they face difficulties to get to know one another and develop a solid relationship:

“...working in a distributed team is so different from working in a co-located team. We have not met some [team] members in person, and we do not know the face of several [team] members.” —P25, Scrum Master.

Many participants reported that keeping members of a distributed team together is challenging because team members may develop different perspectives to a project while working from remote sites:

“...as soon as you separate team members, they have different ideas, different agendas, and their ability to operate on [the] same plan goes away. You can’t keep them together when they are separated.” —P3, Scrum Master.

In the next subsections, we describe how the Same Team strategy develops team cohesion among members of a distributed team. This strategy (in combination with Same Culture 7.1 and Same Practices 7.2 strategies) aims to bridge socio-cultural distance. Figure 4.4 illustrates the tactics for the Same Team strategy.
4.1. SAME TEAM

4.1.1 Knowing the Team

Members of a distributed team ideally should know everyone in the team. Knowing the team allowed team members to recognise one another and understand the structure of the team with respect to who does what within the team. Without knowing the whole team, the team members faced difficulties in developing a sense of togetherness:

“I have not seen the entire team. So it’s difficult to have a feel about the team members and get to work with them.” —P24, Agile Coach.

When team members felt a strong identification with a team, they worked together closely for the success of the project. Conversely, when team members do not know each other, they found it difficult to develop cohesion within the team:

“We used to work together daily, but we did not know each other. I did not have a face for a name. It’s hard to get the feeling of a team when you don’t know who you are working with.” —P9, Developer.

Participants suggested that right from the start of a project, every team member should recognise one another and understand the structure of the team in order to be able to develop a cohesive team:
“The context of “who’s who”, “what’s the team structure”, and “whom do we talk to” is important to form a distributed team.” —P18, Business Analyst.

Many participants (P2, P7, P14, P22, P24, P27, P39, P42, P46, P53) reported that teams kept photographs of all the team members on the wall so that they can recognise one another. The photographs of the whole team point out the presence of a wider team which has members across different sites:

“There’s a really simple way to know the faces of our team in Beijing, Bangalore, or Melbourne. In all those offices, we have nice, big photographs of each of the team members up on the wall so that we can see the whole team.” —P22, Scrum Master.

When a photograph of a team member is not available, a picture was drawn just to remind the whole team that a particular team member exists, and is part of the wider team:

“If I didn’t have the photographs, I’d draw little characters for them so that these individuals actually exist and everyone realises whom we are [working] with.” —P2, Agile Coach.

Knowing each other in a team developed cohesion between team members who were working together from different physical sites. When members of a distributed team knew one another, they may easily engage in casual conversations (s 5.1.3) with one another when necessary arises:

“The team needs to know that they are not separated just because they are distributed in different locations. It is so important for a team [that is] working together on a daily basis to know each other so that they can interact without hesitation.” —P24, Agile Coach.
4.1. SAME TEAM

Cohesion develops within the team when team members know each other from working together on a daily basis. Team cohesion fosters social communication between team members (s 8.1.1), which in turn builds trust between them.

4.1.2 Engaging with the Team

While working on a project, every member of the team gained the reputation as a valuable team member and initiated effort to build solid relationships with the whole team in order to foster a sense of belonging and a feeling of togetherness within the team. Team members engaged with the whole team with a positive attitude to foster collaboration (s 8.1.3), to share expertise (s 8.1.2), or to provide feedback (s 8.1.5), in order to work together as one team.

Although team members across different sites of a distributed team worked together daily on a common project, they faced difficulties to build cohesion within the team. Without face-to-face interaction and video communication where team members can see one another during conversations, they could not build rapport with one another:

“All the team members have not physically met. We have only had conversations over the phone, and we don’t do video conferencing. We’ve not really built the rapport with each other.” —P25, Scrum Master.

Members of a co-located team work in close proximity with one another, thus they can engage in frequent social interaction which in turn helps to build trust between them (s 8.1.1) and foster cohesion within the team. When team members are separated across different sites, however, it can be difficult to engage with one another because team members often communicate for formal purposes rather than for social purposes:
“...we have formal conference calls, but we don’t get to bond with them without social conversations.” —P12, Developer.

Despite regular formal meetings, team members were not able to develop solid relationships and engage with one another without social communication. When team members were not engaged with the team, they become rather passive in discussions and did not wish to contribute to the success of the project:

“Meetings were happening without any purpose, without any useful discussions, and without any active participation from the team.” — P43, Agile Coach.

A senior manager reported that several team members were not willing to engage with the whole team for knowledge sharing or collaboration on project activities. Without the willingness for sharing knowledge or expertise (s 8.1.2) and collaboration (s 8.1.3), these team members face difficulties to build trust within the team:

“There are some individuals who do not engage with the wider team and feel content to sit back and do the bare minimum of what their job demands of them. They often lack willingness to share ideas with the rest of the team.” —P20, Senior Manager.

Several participants (P9, P20, P30, P32, P43, P55) reported that some development team members did not engage with customers. Overall communication between development team members and customers was affected when even a few of the development team members failed to engage with the customers:

“...[development] team members were not engaged with the customer, especially the junior members in the team have never spoken to the customer.” —P32, Senior Manager.
Without effective communication with customers, the development team members did not strive to deliver business value to the customers:

“The team never prepared properly for the demos, and [the demos] weren’t business meaningful to anyone.” —P30, Agile Coach.

Customers, in turn, did not treat the development team members with respect when there was a problem in the software. Without rapport between the development team members and the customer, there was no respect and trust between them:

“We did not have a working relationship with [customers]. When we were delivering the first release of the software, customers shouted at us because it did not work as expected.” —P9, Developer.

Solid relationships within the teams is crucial for the success of the projects. When there is a team relationship that bonds them together, the team members strive to perform as a whole team and provide value to customers:

“We want them to feel that it is one team. The team is working together for the same project. So, we have to build mateship and friendship which are essential for a team.” —P21, Scrum Master.

To build solid relationships with team members, teams used Wikis to share personal information such as family background, interests, and hobbies with everyone in the team. Sharing personal information with team members fosters meaningful interactions between them:

“We have a team-place in the Wiki where we share some moments of our personal life which will help us move forward in our professional interaction.” —P24, Agile Coach.
Proper use of communication technology can help team members to engage with one another. Although team members across sites could not meet face-to-face, the use of video-conferencing during formal meetings (s 5.1.1) helps to foster interaction between team members:

“... ensure that the entire team is getting fully engaged with the customer. To begin with, organise stand-up meetings with video conferencing.” —P13, Scrum Master.

Participants reported that team members often socialise after formal meetings (s 5.1.2) or during cross-site visits (s 6.1.2) in order to build rapport with one another. When team members show interest in social communication (s 8.1.1), trust develops between them:

“... use all the opportunities during remote meetings or face-to-face visits to have a social conversation with them. You have to learn to make friends with the people you’re working with.” —P11, Scrum Master.

When local members meet remote members during cross-site visits (s 6.1.2), they spend time together outside normal working hours in order to get to know them and build a solid relationship between them:

“... we spent lots of good time with them. Some of them were keen to see our village life [in India], and willing to come to our house.” —P26, Agile Coach.

Local members of a team often bring the remote members out for dinner and take them sight-seeing. The interaction during such social activities fosters a solid relationship between them:

“... we often went out with them to the restaurants and beach.” — P29, Scrum Master.

Team members interact on a personal basis before formal meetings (s 5.1.2) and when team members travel to a different site (s 6.1).
4.1. SAME TEAM

4.1.3 Establishing Shared Vision

A project vision is the customer’s inspiration for the project which translates into project goals for the development teams. Sharing the vision and techniques for realising the vision developed an understanding of the business value that a project brings to the customer.

Without a shared vision between the customers and the development teams, the team members could not realise the importance of the project to customers. The team members across the different sites may not work towards achieving the same goals when the vision of the project is not clear to them:

“Without knowing the project vision, the team doesn’t function as one team but [rather] works on the project without a direction. There’s no way for them to know the value of the project to the customer.” — P34, Agile Coach.

Every member of a team gots to know and understand the shared vision of the project, and plan out strategies for achieving success in the projects:

“We get team members from different sites to understand the goals of the project, and the ways to establish velocity and achieve quality in the project.” — P3, Scrum Master.

At the start of a project, teams worked closely with customers in order to understand the project vision clearly and plan out the efforts required to complete the projects successfully:

“We work with the clients and get an understanding of the entire vision for the project. We get to know the complexity [of the project] and the efforts required from team members at both locations.” — P16, Scrum Master.
Senior managers often provided teams with complete and clear information about the project vision so that the team members can understand the customer’s purpose for carrying out a project. When managers fail to explain the project vision, however, the teams faced difficulties to align with the customer’s goals for the project and to deliver business value to the customer:

“*When managers fail to explain the grand vision of the project, the team members are not in a position to deliver value to the customer.*”
—P53, Agile Coach.

Several senior managers (P20, P28, P48) reported that management in the organisations realised the importance of sharing project vision with the teams. With a clear vision of a project, the team members knew their responsibilities towards the project:

“...we take every step to explain the [project] vision to the team so that everyone supports the idea and understands where we are heading, what we need to be doing, and what are the deadlines.” —P48, Senior Manager.

Senior managers placed a lot of importance on team members acquiring consistent information on the project vision across the different sites. A shared vision of the project ideally should develop a shared understanding of the project among members of a distributed team:

“*We spent a lot of time getting the business context and developing a shared understanding of the high-level visions of the project across every location of the team.*” —P20, Senior Manager.

In addition to explaining the project’s vision during formal meetings, sharing the vision in the project’s Wiki allowed every team member to access the same information at any point of time:
4.1. SAME TEAM

“*We wanted everyone to understand clearly the customer’s vision for this project. So, we got it written and posted it in the shared Wiki. We also discussed about it thoroughly with the team members from both locations.*” —P28, Senior Manager.

Participants reported that teams focused on developing an effective collaboration with customers through a shared vision and realising the common purposes for working together:

> “Our clients know that we are fully aligned with them, we share their vision, and their success is our success.” —P14, Developer.

A shared project vision fostered close collaboration between team members across sites (s 5.1.5), for achieving the goals of the project.

### 4.1.4 Team Recognition

Traditionally, managers carry out evaluation to measure individual performance against measurable objectives in order to provide feedback and recognition to the individual. Managers, however, must recognise how the whole team performed in a project rather than how an individual performed his job in a team.

Participants reported that team spirit was dampened when managers failed to recognise the overall performance of a team for the team-oriented software development activities:

> “When the managers decided to evaluate individual performance of the [team] members, it created a very negative impression in the minds of the team members.” —P39, Agile Coach.

Due to the evaluation of individual performance, the team members were competing with each other instead of complementing and supporting one another:
"...because managers are measuring how an individual performs in the team and not how the team performs in the project, so team members are being political in the sense that they seem to be your friend but they are stabbing you at the back." —P6, Developer.

Participants suggested managers must observe and recognise the performance of the whole team in order to develop a sense of togetherness within a team:

"The team is the core of the project. It is most important for them to feel as one team in order to manage the project successfully. So, we always look at the performance of the team rather than [performance] of each individual in the team." —P29, Scrum Master.

Participants also suggested managers also must recognise the overall performance of the whole distributed team dispersed across the different sites rather than the performance for each separate site of a distributed team:

"The team is distributed but there's only one whole team. We did not do any measure per individual or per location. Everyone works as one team, and there is one team performance. It is not a good approach to say that one side of the team has better performance than the other. If there's a problem, both locations need to resolve it together." —P7, Agile Coach.

Several senior managers (P28, P42, P45) suggested the performance of the whole team must be measured for team-oriented activities in order to point out to the members of a distributed team the importance of working together as a team:

"When we advocate that team members should work together and teamwork is important, then we should measure team performance [but] not the individual performance." —P42, Senior Manager.
When a particular sprint was not successful, the senior manager did not attempt to find fault from a particular member of the team but rather held the whole team responsible for the outcome:

“When a sprint failed, we don’t bash an individual [but] we generally say, “The team didn’t perform well, and that’s why the sprint failed.”
We value teamwork.” —P45, Senior Manager.

Similar, in order to encourage teamwork, senior managers recognised and rewarded the whole team for an achievement:

“We reward the entire team for an accomplishment. Everyone gets the same form of reward such as movie tickets or gift hampers.” —P28, Senior Manager.

In order to develop cohesion between members of a distributed team, managers gave due recognition to the performance of the whole team for the team-oriented software development activities.

4.2 Discussion

A large number of studies [7, 66, 90, 119, 166, 311] have reported the impact of team cohesion for the success of a project. Cohesion is seen as strong predictor for the performance of a team [29, 115]. Hence, cohesion has been included in several models of team, teamwork, and team performance [159, 166, 258]. Though team cohesion has been given different definitions [115], one comprehensive definition of team cohesion has been proposed as “an individual’s sense of belonging to a particular group and his or her feelings of morale associated with membership in the group” [32]. Bollen and Hoyle [32] contend that without a fundamental sense of belonging, team members would not engage with their cohorts.
CHAPTER 4. THE THEORY OF ONE TEAM

With cohesiveness as the central concept that affects performance of a team, several studies [246, 274] point out strengthening the cohesion is important for the success of distributed teams in Agile software development. Fiore et al. [90], however, reported cohesion may be difficult to develop in distributed teams because there are simply less or no opportunities for team building activities within the team. When team members are new to each other, a distributed team ends up as “a community of strangers” [119] whom are working on a common project.

Chin et al. [45] suggest the issue of cohesion has become critical for virtual teams where technology replaces face-to-face communication. Gupta and Fernandez [119] discuss team presence awareness across sites such as who is online, who is working on what, and whom to contact is critical for team members to work together in cohesion. From our study, we found team members develop a sense of belonging to a distributed team when they know one another in person. When team members have not met, however, the teams keep photographs of each team member on the wall so that they can recognise one another.

Jones and Harrison [166] reported that members from a cohesive team engage with one another in the team-oriented tasks in order for the project to be successful. A highly cohesive team has been found to provide value to customer and meet project schedule, budget, and quality. Our findings show team members engage in social communication with the team in order to build a solid relationship that bonds them together and motivates the team to be successful in the project. The opportunities for social communication happen during casual meetings using communication technologies (s 5.1.2) or when team members travel to different sites (s 6.1).
4.2. DISCUSSION

On investigating characteristics of Agile teams that relate to team cohesion, Whitworth and Biddle [311] reported that a lack of shared vision of what was going on within a project was related to lack of team cohesion or togetherness. A shared vision of a project allows a team to understand the project goals and requirements to complete the goals. Hence, the shared vision is an invaluable source of cohesion and critical for the performance of the team. We found senior managers understood the importance of a shared vision for distributed teams and contributed to developing team cohesion through articulation and effective sharing of project vision with the entire team.

Salas et al. [257] reported that the overall performance of a team improved when team members engaged with one another while carrying out software development activities. When an unanticipated change happens in a project, the team relies on the team cohesion to adapt to the changes rapidly and effectively [7]. In a team that lacks cohesion, however, team members are less likely to help out when workloads get heavy or when a team member experience conflict with another [126]. Our study reported senior managers help to develop the feeling of togetherness through recognising overall team performance instead of individual performance for the team-oriented software development activities. With the feeling of “teamness” [126], a distributed team works together effectively while sharing a common workspace (s 7.2.3) and engaging in real-time collaboration (s 5.1.5).
Chapter 5

Bridging Temporal Distance

5.1 Same Time

Same Time strategy aims to promote synchronous communication in order to bridge temporal distance.

Co-location provided team members with opportunities to get instant feedback from one another, especially the customer. Participants reported team members faced difficulties in getting instant feedback when the team is separated across time zones:

“...the purpose of co-location is instant feedback. When we remove the co-location, getting instant feedback is almost impossible.” —P32, Senior Manager.

Many participants discussed the difficulties of working together when teams were separated across different time zones, especially for obtaining quick and timely feedback on project tasks:

“Since the day we started the project, it was especially hard to get feedback from the customers.” —P9, Developer.
“Less synchronous communication in the team leads to less instant feedback, or feedback was not timely.” —P28, Senior Manager.

When team members needed clarifications, they may have to wait for a long time before feedback can be received from the remote site. Delay in receiving crucial feedback often reduced the productivity of the team:

“We have to wait 8 hours or 10 hours in order to get some response from the team members [in the USA], [and] that delay seems to reduce the productivity of the team.” —P29, Scrum Master.

Several participants (P1, P4, P5, P7, P43, P52) suggested team members should not be dispersed across time zones in order to maximise the time overlap between the different sites of a distributed team:

“The best way is to have the whole team in the same time zone instead of splitting these teams members across different time zones. The time zone is painful!” —P5, Agile Coach.

“When [a team is] distributed, it is always better that [team members] are closer together, in terms of time difference.” —P4, Agile Coach.

In the next subsections, we describe how the Same Time strategy promotes synchronous communication among team members who are dispersed across different time zones. This strategy provides tactics for working together at the same absolute time. Figure 5.1 illustrates the tactics for the Same Time strategy.

5.1.1 Planned Formal Meetings

A planned formal meeting is a scheduled, formal meeting that a team organises for a specific purpose. Formal meetings regularly happen within project teams in the form of daily stand-up meetings, planning meetings, sprint review, sprint retrospective meetings, or product release meetings.
5.1. SAME TIME

Each of these meetings serves an important purpose for software development activities. For example, the main purpose of a daily stand-up meeting is synchronisation between team members, and the main purpose of a review meeting is the demonstration of new features to a customer.

Many participants reported organising formal meetings for a distributed team is challenging because team members across different time zones do not have a time overlap during normal working hours. Teams, however, found ways (s 5.1.4) to organise the formal meetings in order to promote synchronous communication between team members:

“When teams are distributed between USA and India, you have minimal time overlap for a daily meeting [but] you are definitely better off doing it than not doing it.” —P2, Agile Coach.

Participants reported that team members put in a substantial effort to interact with the entire team during formal meetings. For example, some team members have the answers to the Scrum questions in writing (P3, P7, P53) or prepare a list of critical questions to ask the customers (P4, P15, P19, P39, P55) before the stand-up meetings:
“We communicate very well with everyone during every formal meetings because we understand that we do not get opportunities to communicate with them throughout the day. We put in a lot of effort so that the formal meetings are successful.” —P40, Developer.

In order to get the development team members, customers, and senior managers to meet and talk to one another, teams organise formal meetings using video communication:

“...we want the team members to have some sort of face time between them and with clients. We have to organise formal meetings with all stakeholders so that everyone will be present, and we get to know who is in the team [so that] communication gets much easier later on.”
—P16, Scrum Master.

A key benefit of the formal meetings is that team members get instant feedback from customers (or customer representatives):

“During the stand-up [meetings], client representatives will be there, they certainly respond to every question we ask them. I feel these official stand-up meetings are more effective than emails to communicate with clients.” —P15, Application Tester.

Teams conducted retrospective meetings after each iteration to assess the level of success of the iteration against the iteration and projects goals. The project team used the experience and knowledge acquired during the course of an iteration for improving processes and practices that are in place within the team:

“After each iteration, we are doing the review meetings and retrospectives. Based on the feedback from team members, we’ll be taking corrective actions and analysing whether the project is going in the right way and whether it can be completed in time. So, we plan [future] iterations and projects accordingly.” —P23, Quality Analyst.
Team members shared technical expertise or domain knowledge with everyone in the form of live presentations:

“We brought about an idea that once in a week, a team member has to identify a topic of his interest and share it with the team. Once a week, at the end of the daily stand-up meeting, we have a 15 minutes presentation where team members share their own technical expertise with the entire team.” —P34, Agile Coach.

When teams engaged in frequent discussion with one another or with subject matter experts, the team members gained sound knowledge on the Agile practices, technologies, and software development processes used for a project:

“We introduced bi-weekly technical huddle sessions to share technical knowledge on what we are doing and what are the things blocking our work. We have Skype conversations with the on-site team and client, and discuss things with subject matter experts. After every session, the communication certainly went up a fair bit, collaboration with client improved, and we understood better what we were developing.” —P38, Business Analyst.

Formal meetings provide team members with good opportunities to interact with one another to the share impediments and the progress of a software development project, and discuss potential solutions for their problems. Most importantly, team members get quick feedback from customers and share technical expertise or domain knowledge in order to improve on Agile practices, technologies, or software development processes. Hence, communication (s 5.1.3) and collaboration (s 5.1.5) improve in the team.
5.1.2 Planned Unconference

A planned unconference is a scheduled, informal meeting that a team organises in order to facilitate interaction between team members. A team often plans for an unconference prior to a formal meeting and allocates a short period of time for team members to interact with each other on a casual basis:

“We need to have some personal time with other team members. We have 15 minutes before daily meetings to speak freely to each other in the team [so that] team members can understand and value the other team member’s life, [and] build rapport with all team members.” —P24, Agile Coach.

Participants reported that casual conversations during the planned unconference fosters solid relationship between team members and contributes to building a successful team:

“... the first 15 minutes was open time, and we could talk about anything we want, share personal stories, and be part of the fun conversation. And, that's when we started seeing a very strong team bonding.” —P1, Developer.

Casual conversations in a planned unconference provide team members with opportunities to understand one another because they get to discuss issues, concerns, or project related matters with one another, and further understand information that has been shared with them:

“...get team members to talk to each other so that we can understand the information and knowledge [that] we share, and avoid misunderstandings in the team.” —P53, Agile Coach.

Participants (P1, P2, P11, P15, P24, P34, P38, P46) reported that distributed teams emulate co-located teams in which team members regularly engage in casual conversations to share personal matters with everyone:
“Before retrospectives [meetings], we make sure that there’s always time allocated for casual conversation. Somebody will say, “My son got an award”, or “I went on a vacation and I did hang-gliding”. It just creates a better environment for the team because distributed members are not in front of each other.” —P46, Agile Coach.

Participants reported team members engaged in active participation during the formal meeting that follow a planned unconference because casual conversation during a planned unconference helps to develop cohesion among team members:

“The team as a whole definitely benefit from having the casual meetings just before the formal meetings. Team members feel elated after the casual conversation, and they engage in discussion and willingly share ideas or opinion with everyone during the formal meetings.”
—P42, Senior Manager.

The main purpose for a distributed team to organise a planned unconference is to emulate a co-located team in which team members frequently participate in social communication (s 8.1.1).

5.1.3 Spontaneous Conversations

Spontaneous conversations are casual conversations that team members initiate outside planned meetings, such as one-on-one phone calls or personal video conversation. An individual often initiates conversations for acquiring information or personal interest, such as to request for a clarification or just to find out how someone is getting on:

“...when we need some information from them, we just pick up the phone and call them [so that] we immediately get the information that we need.” —P37, Developer.
...some of them would call me over the phone just to ask how am I doing.” —P24, Agile Coach.

Participants reported that spontaneous conversations happen regularly for co-located teams such as during breaks or even at the corridors of an office building. When team members initiate conversations with remote members, a sense of togetherness develops between them:

“We had casual conversation over phone with distributed team members in the way we meet with co-located team members at a tea room or at the corridors. We have regular conversations on and off to catch up with project matters or just simply for other matters. We had the sense of working together as a co-located team mainly because we had these regular conversations with one another.” —P54, Developer.

Spontaneous conversations were important for distributed teams because team members can get to know one another and build a solid relationship within a team:

“We just need to have a casual conversation and get to know each other. We need to keep regular conversations right from the start of the project till the end of the project [so that] there’s a good relationship within the team. This is important for building a team.” —P21, Scrum Master.

Many participants (P1, P2, P3, P6, P7, P21, P27, P24, P37, P40, P54) pointed out that team members realise the importance of casual conversations for building a solid relationship within a team:

“When teams members are motivated to make this project successful, they often find time to engage in casual conversation, such as talking over the phone, [in order] to get to know [others] better.” —P6, Developer.
Several participants (P18, P20, P28, P29, P38) reported that a significant benefit of spontaneous conversation is effective knowledge sharing between team members. During casual conversations, team members tended to share experience and project critical information with one another:

“Most information sharing doesn’t happen during the meetings, [rather] it happens incidentally during one-on-one phone calls or instant messaging.” —P20, Senior Manager.

When technical infrastructure and communication tools were available across sites, the team members engaged in spontaneous conversations whenever needs arise such as for requesting information:

“...we have the infrastructure to support them to make a phone call or video conference when they need to get in touch with a remote member.” —P27, Scrum Master.

Team members kept a regular and continual communication (s 8.1.4) with one another on a professional and social basis when opportunities arise, or deliberately create opportunities to interact with team members to promote synchronous communication:

“We have to keep talking and keep continual interaction with team members. Whether it’s in [formal] meetings, whether it’s talking to team members one-on-one, or when pair-programming.” —P20, Senior Manager.

Spontaneous conversations provide opportunities for team members to engage with the team (s 4.1.2) and to share knowledge (s 7.2.2) between team members across sites.
5.1.4 Shifting Working Hours

Time-zone differences among the sites of a distributed team affects opportunities for synchronous communication between local and remote team members. One participant highlighted the impact of the substantial time-zone difference between India and the USA:

“We are in central time zone [in the USA]; we were off by 12.5 hours [with India]. We are often holding up our daily calls.” —P1, Developer.

In order to create a time overlap for synchronous communication with the whole team, such as attending formal meetings, some team members stretched their normal working hours:

“We stretch our working hours till late evening [because] this is the only way we will be able to communicate with them.” —P26, Agile Coach.

Stretching working hours created frustration among team members for working long hours:

“We sometimes have to work long hours so that we can be [available] online [in order] to ask the US developers some technical queries.”
—P12, Developer.

Team members were unhappy spending personal time for attending stand-up meetings:

“We used to have our stand-up meetings in the night [but] it was ridiculous [because] we are actually taking our personal time from 9:30 to 10 pm everyday for dialling into stand-up meetings.” —P18, Business Analyst.
5.1. SAME TIME

Some organisations offered flexible working hours where team members may shift their normal working hours to suit their team:

“The flexible working time is employed to everyone in our organisation, but how flexible it is depends on our immediate manager.” —P44, Product Owner.

“...we have team members who come in to work in the afternoon and they work till late night.” —P4, Agile Coach.

Participants reported that they understand the importance of shifting their working hours in order to create a time overlap with remote sites:

“We actually do a video conference every day at 7 o’clock in the evening with all the team members and customer representatives [because] our objective is to work together as one single team.” —P13, Scrum Master.

“The call happens late in the evenings [for India] where it is early morning for them [in the USA] [but] we know that it is always good to have a conversation with everyone [in order] to understand the technical aspects to the software development.” —P17, Developer.

Substantial time-zone differences between sites of a distributed team causes difficulties for the real-time collaboration (s 5.1.5) and spontaneous conversation (s 5.1.3) between team members. Shifting their working hours, however, creates the time overlap between team members for synchronous communication. Several participants (P5, P16, P18, P28, P40, P46) describe how team members from different sites take turns to shift normal working hours (s 7.2.4) in order to share the difficulties faced due to time-zone differences across sites.
5.1.5 Real-time Collaboration

Real-time collaboration is using the Internet and technology for working together synchronously with team members across different sites. Participants reported teams benefit from real-time collaboration with respect to shared knowledge, improved communication, and shared understanding:

“There’s a lot of gains out of close collaboration at real time with team members across different sites. We get new ideas from everyone, improve communication, and improve shared understanding in the team.” —P22, Scrum Master.

Teams must not rely on written communication only but rather use different kinds of communication technologies in order to collaborate effectively:

“If you’re relying on email only, or Instant Messenger only, or anything that’s just a written medium, your ability to collaborate is much lower and the ability to misunderstand is a lot more.” —P22, Scrum Master.

Participants reported phone or video communication is preferable to written communication, such as internet chatting, for real-time collaboration. Voice communication promotes social interaction between team members and reduces misunderstanding during the collaboration:

“It is important to get team members to talk to everyone at a person-to-person level. When they are talking, there’s less chances for misunderstandings. So, make sure the technology supports distributed teams to have voice communication.” —P31, Agile Coach.

When team members used phone communication, however, it was difficult to gauge their level of interest and comprehension on a discussion:
“With a distributed team, you’re on one side and you’re telling the
team on the other side what to do. And, you’re hoping that they
are getting it, you’re hoping that they are not surfing the net while
listening to you. If they’ve switched off, you don’t even know it!”
—P44, Product Owner.

Phone communication posed challenges for team members to partici-
pate actively in group discussions:

“It is much harder [to communicate] in a group meeting using the
phone. If you’ve got conversations going on in a room full of people,
then there’s only one phone in the middle, you can’t hear what people
are saying.” —P6, Developer.

Many participants reported teams often combine several kinds of syn-
chronous communication technology with collaboration tools for real-time
collaboration. Teams, however, particularly preferred video communica-
tion for effective collaboration with remote team members:

“We have a mix of communication tools because we know that we
can’t always use one single communication tool for collaboration. We
often use video conferencing with collaboration tools. Using these
tools, we work together on a daily basis to discuss requirements or
user stories which are going through constant changes.” —P55, De-
veloper.

Video-conferencing captured visual aspects of communication such as
head movements and hand gestures which were important for effective
communication:

“We found the visual aspects in communication is so important that
we encourage teams to have video conferencing [during real-time col-
laboration].” —P1, Developer.
Participants reported video communication improved team interaction and subsequently resolved conflicts between team members:

“We found that [video communication] has really added value to the team because we can talk to other members in a different site and see his or her face expression too. Slowly we were able to resolve a lot of conflicts that arise when working together because we were able to see while talking to them.” —P26, Agile Coach.

Pair programming provided means for knowledge sharing and enhanced learning in distributed teams through frequent interaction with one another. When distributed teams adopted pair programming using collaboration tools, the time difference affected the duration for remote pair programming:

“We have tools for distributed pair programming, but we do it for a shorter period of time because of time difference between India and [the] USA. We normally get a junior developer to pair with a senior [developer].” —P12, Developer.

Developers mutually shared technical expertise and experience during pair programming, and acquire knowledge and skills from one another:

“...remote pair programming has been useful because knowledge transfer between the different sites increases our understanding of the project and I often get to learn new skills from my counterpart. We do remote pair programming for half a day, everyday.” —P40, Developer.

With the main focus on synchronous communication, real-time collaboration fostered instant feedback between team members:

“When we do a code review or discuss requirements or share an important concern with remote members, we know that we can get a quick or immediate response from them.” —P55, Developer.
Many participants reported teams benefited from real-time collaboration because team members get to support one another in terms of sharing knowledge, expertise, and experience in software development:

“Collaboration between all the members in the team has been the most important success factor for this project. We always work together, and we knew weaknesses and strengths of each other. So, we were able to adjust to one another.” —P54, Developer.

On a daily basis, members of a distributed team worked together with one another (s 7.2.3). Real-time collaboration, however, required team members to be available at the same absolute time and use synchronous communication technologies. Participants (P18, P20, P24, P30, P31, P32, P37, P40, P50, P55) reported time overlap between the different sites of a distributed team supported real-time collaboration. When there was no time overlap during normal working hours, team members shifted their working hours (s 5.1.4) in order to create time overlap which allowed them to work together synchronously.

5.2 Discussion

Despite a wealth of literature on the impact of temporal, spatial, and socio-cultural distances on distributed software development [8, 39, 55, 125, 149, 224], there is, however, a paucity of studies into the impact of these distances on Agile software development (e.g. [150, 153, 185]). In this section, we focus our discussion on the impact of temporal distance on distributed software development in the light of related literature.

While exploring the assumed benefits of distributed software development, Conchuir et al. [55] reported that distributed teams fail to leverage time-zone effectiveness. A popular assumed benefit is the follow-the-sun development where organisations decrease cycle-time with an increased
number of working hours during a day by having developers located in different time-zones. They reported that follow-the-sun was not used for development activities, but rather for other activities, such as testing. When team members are faced with reduced collaborative time and unusual working hours, time-zones are not considered a benefit for distributed teams.

Holmstrom et al. [149] discuss in detail how temporal distance challenges everyday communication within development teams. They reported that distributed teams face delay of responses that frustrated team members and affected team performance. While asynchronous tools such as e-mails and Wikis were critical for communication and coordination, the use of these tools over temporal distances increased the response time. A solution proposed to manage time zone differences was to have the teams distributed between no more than two sites.

Through a review on global software development, Noll et al. [224] summarise several barriers and solutions for distributed team. A widespread problem with temporal distance in those studies was the delay in responding to inquiries. They reported that the primary solution to overcoming temporal distance is using synchronous communication technology that provides as much of the “in person experience” as possible.

Carmen and Agarwal [39] strongly encouraged the use of synchronous communication by teams across time-zones. They reported that synchronous communication helps to resolve miscommunications and misunderstandings, and prevent unnecessary conflicts within the team. Unlike asynchronous communication that delays or complicates problem resolution, even a brief conversation can quickly clarify the problem or provide team members with a clear situation of the problem. From our study, we found teams organised planned, casual meetings for a short period of time, prior
to formal meetings. Casual conversations provided team members with opportunities to get to know one another.

Cummings et al. [59] carried out a study to investigate the impact of spatial and temporal boundaries on coordination delay. They reported that temporal distance change the synchronicity and timing of activities between the different sites of a distributed team. With a decrease in overlapping work hours, they found a significant increase in time for responses and waiting periods. While temporal distance reduced the flexibility of team-oriented tasks, the coordination delay affected project performance. Overall, they found that challenges from temporal distance are more difficult to overcome with communication technologies than spatial distance. From our study, we found distributed teams either adopt flexible working hours or extend the normal working hours in order to create time overlaps with remote members. Several participants were not supportive of attending formal meetings at the expense of personal time, especially late in the evening. Most of them, however, clearly understood the need for attending such formal meetings. In order to be fair with team members from both the local and remote sites, teams share the pain of shifting the normal working hours (§ 7.2.4) either for each iteration or each month.

Agerfalk et al. [8] found distributed teams are forced to rely on asynchronous communication technologies due to temporal distance across multiple time zones. Asynchronous communication, however, does not support the team in acquiring quick responses during normal working hours. In order to initiate contact with cohorts at other sites and to provide quick responses for inquiries, teams shift their working hours to create the temporal overlap with remote members. From our study, we found shifting normal working hours to promote synchronous communication allows team members to engage in spontaneous conversations in order to acquire instant feedback from one another.
An important tenet of Agile software development is frequent interaction between the development team and the customers in order to obtain instant feedback from customers or share knowledge within the development team [51]. Herbsleb [125], however, reported temporal distance causes much less communication and less effective communication. Temporal distance limits informal communication and limits the degree to which tacit knowledge is shared among team members. Batra [21] suggests Agile practices may need to be modified for global software development, while Holmstrom et al. [150] suggest distributed teams have to be flexible to achieve overlap with remote members. We found most information sharing happens during spontaneous conversations with one another. Teams shift working hours to engage in real-time collaboration such as distributed pair programming or collaborative code reviews. Real-time collaboration fosters knowledge sharing and promotes fast response within the team.

Through a case study that investigates the use of Agile methods in global software development, Hossain et al [153] reported that Agile methods support the distributed teams in being successful in their projects. With a team distributed between Australia and Malaysia, there was a significant six hours time overlap between these sites. They reported that daily stand-up meetings help to minimise the possibility of coordination breakdown, while sprint reviews increase project visibility and transparency. Agile practices such as code ownership help project stakeholders to maintain a common set of standards throughout the development. Overall, Agile practices helped to overcome temporal distance. We found that despite the temporal distance, teams adopt Agile practices such as daily meetings in order to increase visibility into the project status. Teams with a substantial time overlap between the different sites practice distributed pair programming daily for half a day, and engage in real-time collaboration to prevent coordination breakdown.
5.2. DISCUSSION

Paasivaara and Lassenius [231] reported findings from a multiple-case study of four globally distributed projects that were using Scrum. They suggest daily Scrum meetings were the most useful practice for distributed teams to share information and coordinate work across different sites, to recognise possible impediments early in the project, and to facilitate subsequent asynchronous communication after the meetings. Our findings confirm the importance of Scrum ceremonies in general and daily Scrum meetings in particular for promoting synchronous communication between different sites on a daily basis. Scrum meetings provide avenues for distributed teams to get instant feedback from customers and to get to know everyone in the team.
Chapter 6

Bridging Spatial Distance

6.1 Same Space

Same Space strategy aims to promote temporary physical co-location in order to bridge spatial distance.

Face-to-face communication is the most efficient and effective form of communication for conveying information to and within Agile teams:

“Face-to-face communication is the best!” — P5, Agile Coach.

Face-to-face communication facilitates knowledge sharing and fosters cohesion between members of a distributed team:

“We need to have face-to-face conversations to share knowledge with the team and to get a shared understanding of the project.” — P22, Scrum Master.

“Face-to-face communication is really important to build up rapport between team members. When we work in a distributed team, regular face-to-face meetings can make a huge difference to the project.” — P11, Scrum Master.
Participants reported that frequent face-to-face communication between team members is crucial for the success of a project. Hence, team members travel to remote sites and work in close proximity with the rest of the team members in order to facilitate face-to-face communication:

“There’s something about people that they need to meet face-to-face and smell one another [in order] to work together as one team.” — P3, Scrum Master.

“The key thing for distributed Agile teams is to get team members meet face-to-face on a regular basis. So, for all projects with distributed teams in our organisation, team members were travelling back and forth between different locations to meet the whole team.” — P40, Developer.

In the next subsections, we describe how the Same Space strategy promotes temporary physical co-location in order to bridge spatial distance. This strategy provides tactics for producing temporary physical co-location through movement of team members from different sites to a same physical space. Figure 6.1 illustrates the tactics for the Same Space strategy.

![Figure 6.1: Tactics for the Same Space strategy](image)
6.1. SAME SPACE

6.1.1 Inception Workshops

Agile software development projects typically started with an interactive series of structured inception workshops for project chartering. Customers, subject matter experts, and several other key stakeholders met face-to-face during the inception workshops to discuss the project contexts and the customer’s vision for the project:

“We usually kick off projects with a series of workshops [where] we discuss in details the business context, organisational context, the customer’s motivating factors, and the project trade-offs such as time, quality, scope, and budget.” —P20, Senior Manager.

Subject matter experts, or individuals who possess specialised knowledge or skills required for a project, such as implementation experts (developers) and quality experts (testers) provided technical advice to the stakeholders:

“...a couple of subject matter experts attend the inception workshops to give some technical advice on the project.” —P27, Scrum Master.

Participants in one project described inception workshops that lasted for two weeks where teams carried out comprehensive discussions with respect to project goals and vision, business benefits, costs, schedule, success measures, technology assumptions, and rules of engagement. The inception workshop concludes with a shared understanding of the project and prepares the development team for software development activities:

“...we met up with the clients, talked about the project goals, their vision of the project, and what is the scope of the project. We decided on the communication plan, the overlap hours between the different locations, how the requirements are going to be passed to the project team, who are the Product Owners, who are the stakeholders, and who is going to do the sign-offs.” —P11, Scrum Master.
A full-fledged team would be formed based on the project context, and the project manager seeks commitment from the customers to work together on the project with the development team:

“We have an inception workshop involving several key stakeholders before the start of a project. We later form the team according to the [project] needs outlined in this workshop.” —P36, Quality Analyst.

Team members who attended the inception workshops established direct communication and close collaboration with the customers, and developed a shared understanding of the project:

“One ideal time to meet the customers is during [project] inception. An inception is so critical to the success of the project [because] that’s when customer begins collaboration with the team.” —P2, Agile Coach.

With opportunities for casual conversations with customers, the team members got to know the customer and started to build a solid relationship between them:

“...it was easy for me to build rapport and build a good relationship with the customer while we were together at the inception workshop [because] we had plenty of time for conversations outside of work.” —P38, Business Analyst.

Customers understood that full commitment towards the project and close collaboration between development team and customer, determined the success of the project:

“...commitment from the customers is so important for the success of a project.” —P18, Business Analyst.
Participants reported that some team members could not attend the inception workshops because the team members joined the team at a later stage of a project (P29, P30, P34, P43), the budget constraints limited the number of team members who travel to attend the inception workshops (P9, P11, P17, P19, P52), the existing commitments prevented them from attending the inception workshops (P12, P23, P25, P44, P45). Nonetheless, inception workshops provide the initial opportunity for the development team and the customers to develop a shared vision (s 4.1.3) and engage in social communication (s 8.1.1).

6.1.2 Cross-Site Visits

In a cross-site visit, a team member travels to a remote site of the team in order to see and spend time with the team members at that site. One of the reasons for a visit is to get to know team members at remote sites and build rapport with them. A team member who travels to a remote site of a team solely to foster interpersonal relationships within the team is often called a “team ambassador”:

“...we have [team] ambassadors who travel back and forth just to build relationships with everyone in the team.” —P4, Agile Coach.

Many participants reported that development team members, customers and senior managers visited one another at different sites on a regular basis. During the visits, the team ambassador built rapport with others and developed a solid relationship with them:

“When we go over to a different site, we work with them, go out with them, eat with them. In that way, we build this friendship, [and] we can get to know the people who we work with.” —P12, Developer.

Cross-site visits provide team members with opportunities to understand one another in terms of how an individual interacts or works together with others in a team. Participants reported that they were able to
work together as a team when they have started understanding each other after the cross-site visits:

“When we went there, we started observing people and their way of interacting with each other. We mainly wanted to understand the team members and know each other. When we started understanding each other, only then we actually started to work as a team.” —P33, Agile Coach.

While working at the remote site during a visit, the team members frequently engaged in social communication with one another (s 8.1.1). The relationships between the team members improved and they felt comfortable working together as a team:

“When some team members from onsite visited our location, team had more face-to-face interaction and everyone started to feel more comfortable with one another.” —P34, Agile Coach.

“We spent a lot of time outside working hours to get to know them and build a good [team] relationship.” —P26, Agile Coach.

Customers often visited team members at different sites. The bonding established through face-to-face interactions paved ways for a successful collaboration between the customers and the development team members:

“Customers should be willing to travel to India, talk to [development] team members, go out with them and eat the food, and be part of the [team] there. It creates a really strong bond between them, and those personal relationship are stronger than any contractual relationship.” —P20, Senior Manager.

Participants suggested that ideally every team member should get an opportunity to visit the remote site and work in close proximity with the rest of the team members. Hence, cross-site visits were encouraged for everyone in the team rather than for selected individuals only:
“We send somebody from one location to another location for a period of time, and then somebody else will switch with them. So, this way different team members can get to understand what it is like working with a team member from a [different] location.” —P46, Agile Coach.

In some projects, however, managers faced difficulties sending all the members of a team to visit different sites due to budget constraints. Therefore, only senior members of the team visited team members at the remote sites:

“These visits can be quite expensive. So, we get some of the senior members in the team to go there just to have a feel about the team members there. It helps them to build rapport with others at another location.” —P24, Agile Coach.

Besides developing interpersonal relationships, another reason for cross-site visits was to foster knowledge sharing between team members through face-to-face communication:

“The primary reason for the visit would be knowledge sharing. Often somebody at a [remote] location has got some specific knowledge that need to be shared, or somebody wants to learn a specific knowledge available in one location.” —P27, Scrum Master.

Subject matter experts visited team members at different sites in order to provide them with an in-depth understanding of critical knowledge required for the project through face-to-face communication:

“When we are dealing with high levels of complexity and ambiguity in the project, we get the subject matter experts to visit team members in different locations, and explain to them the requirements or specifications in face-to-face meetings.” —P28, Senior Manager.
Visits facilitated effective knowledge sharing through face-to-face communication rather than video or phone communication from remote locations because team members and the subject matter experts gets to interact face-to-face for a longer period of time:

“We often get [subject matter] experts from Australia to visit our site and teach team members here on certain technical aspects the project [so that] knowledge sharing can happen throughout the day during social conversations rather than just during a specific time period. It is so much more effective to learn from them through face-to-face interaction rather than using video conferencing between remote locations.” —P40, Developer.

During cross-site visits, the remote team members became aware of the culture in a different site and understood the working conditions in that site:

“When team members from onsite come [to India] and work with us for a short period of time, they get to understand our difficult conditions and constraints. Those short visits helped them a lot to understand how we were working and the cultural difference between both sites.” —P17, Developer.

While most participants had travelled to remote sites at the start of a project or on a regular basis throughout a project, there were several participants (P6, P8, P9, P17, P23, P25, P29) who did not travel to remote sites for the current project.

Cross-site visits (s 6.1.2) and rotations (s 6.1.3) improve communication through face-to-face interaction, foster solid relationships between team members (s 4.1.2), promote knowledge sharing (s 7.2.2), and create cultural awareness (s 7.1.1).
6.1.3 Rotation

A rotation is in fact a special form of cross-site visit where team members of the same role swap sites. Participants highlighted that the main reason for practising member rotation is to maintain the team structure at both the sites of a team:

“We rotate a developer from India with a developer from [the] USA, or a business analyst from India with a business analyst from [the] USA so that we can maintain the [team] structure at both the locations.” —P13, Scrum Master.

Rotation provided opportunities for team members to engage in face-to-face conversations with members from another site and build team relationship with them:

“We want team members from each location to understand a little bit more about the other location. Therefore, we rotate one another to facilitate face-to-face conversations. This rotation promotes discussion and heightens the level of comfort that they have with the team.” —P37, Developer.

Rotation often provided team members with experience of working with remote members from different cultures. Team members learned from working with one another from different countries or cultures:

“...we get to know how to work with a remote member of the same role but from a different culture.” —P27, Scrum Master.

During pair programming, team members engaged in frequent interaction and shared knowledge with one another. Member rotation allowed the pair programmers to experience pair programming with local and remote team members and to understand one another from both technical and cultural aspects:
“...local developers do pair programming with one another and later during [member] rotation, a local developer pair programs with a remote developer. They get to understand not just the technical aspects but also the cultural aspect of working with one another.” —P16, Scrum Master.

Teams continually rotated team members between different locations so that team members can interact with one another, share knowledge, and maintain a close collaboration:

“...keep rotating team members between different locations so that team members can talk to one another, share technical skills, and collaborate on the project.” —P43, Scrum Master.

A rotation has all the benefits of a cross-site visit (§ 6.1.2) while maintaining the structure of the team at each site. Rotations also provide teams with opportunities for learning cultures of remote members (§ 7.1.1).

### 6.1.4 Coach Travels

The fundamental role of an Agile Coach in a distributed team is to facilitate consistent use of Agile methods and software development processes across the sites of a distributed team:

“If coaching the teams, I make sure that team members at both sites are consistent with the use of Agile methods [and] I want them to be consistent with the processes in software development.” —P2, Agile Coach.

Teams benefited from coaching in terms of streamlining the processes, using suitable communication and collaboration tools, and bringing the distributed team members together as a whole team:
6.1. SAME SPACE

“The team really do expect to have a coach who knows the problems in the development process, gets the whole team together and socially know each other, and brings some of the tools into use.” —P47, Agile Coach.

Distribution dampened team members’ engagement with one another (S 4.1.2). Hence, coaches emphasised the importance of working together as a team and cultivated team spirit in order to improve communication within the distributed team:

“[After] we had the coaching activities, we were able to build a successful team. We could see good communication happening between [team members from] different sites.” —P33, Agile Coach.

Coaches strived to improve interaction amongst team members and to develop a shared understanding across different sites. Coaches, however, found it challenging to conduct effective coaching for team members whom they did not know personally:

“Coaching someone that we have never had a personal connection with is very difficult.” —P7, Agile Coach.

Most coaches (P2, P4, P5, P7, P24, P31, P33, P46, P49, P53) travelled around different sites in order to meet the entire team and established team relationships with team members. The personal interaction and the bonding with team members allowed coaches to continue with effective coaching from remote sites:

“We need to travel as much as possible and meet the team members there. So, I think that initial face-to-face interaction helps me later on to keep on coaching from here [at a remote site].” —P7, Agile Coach.

Understanding the benefits gained from coaching, senior managers continued to support coaches to travel across the different sites:
“We had a coach from onsite who came here [to India] for several weeks. That coaching improved interaction with remote team members because the coach helped them to understand the working style of the remote team members. We prefer the coach from onsite location to visit our team frequently.” —P28, Senior Manager.

Coaches also engaged in teaching remote team members to coach one another so that during the absence of a coach, a team member from the remote site may adopt the role of a coach:

“I am also teaching team members from different locations to coach each other [because] coaching is a continuous effort.” —P49, Agile Coach.

Coaches travelled to different sites of the team so that personal connections can be established between a coach and team members in order to provide effective coaching. When team members needed guidance from remote coaches, they interacted directly with the coaches (s 5.1.3).

6.1.5 Temporary Co-location

Temporary co-location is placing the whole team together in a same physical space for a short period of time. Co-location promoted face-to-face communication between all members of a team and encouraged close collaboration between them. While only the project managers and subject matter experts often represent the development team at the inception workshops (s 6.1.1), the temporary co-location brought the whole team together to work on the project for a short period of time.

Co-location provided every opportunity for team members to develop cohesion (s 4.1) while working together in close proximity with one another:
“...get them all together [and] have them work together, co-located, and it doesn’t even have to be for a long period of time. They need to interact face-to-face and get to know one another. Otherwise, this [project] will not be successful.” —P3, Scrum Master.

When development team temporarily co-located with the customers, the whole project team benefited from the co-location:

“At the moment, we go distributed for various reasons [but] temporary co-location of the whole team with the customer is always useful for the entire team.” —P17, Developer.

When teams practised temporary co-location at the start of a project, the team members who did not participate in the inception workshop (s 6.1.1) got to know the customers, built rapport with them, and subsequently acquired direct and rapid feedback from them on an ongoing basis:

“The idea was to start all together as a whole team here for the first iteration in order to have direct interaction with customer. It was a common practice in our company, and everyone was happy to travel and work co-located for a short period of time.” —P7, Agile Coach.

Temporary co-location has been useful to get to know the whole team, to improve social communication (s 8.1.1) and to foster collaboration between the development team and the customers:

“Whenever we start a new project, we make sure the whole team travels and comes together with the customer, gets to know one another, gets compatible with every team member, understands the initial aspects of the projects, and then we come back. It was amazing, and so we do it for every project.” —P45, Senior Manager.

Participants reported that close collaboration within the whole team started during a temporary co-location:
“Collaboration started to happen when team members were sitting very close to each other. The major positive impact during a temporary co-location was [individuals] started to work as one team.” —P34, Agile Coach.

Co-location developed a solid relationship between team members and promotes shared understanding within the team. Managers seize the opportunity during temporary co-location to develop a shared vision of the project (S 4.1.3):

“There’s a need to spend some face time with client and entire team before actually separating them out. So, I would co-locate a team for the first few sprints of the project [so that] the team is able to build [team] relationships.” —P22, Scrum Master.

Some teams rotated the location for temporary co-location between the different locations of a team. Rotation of the location for temporary co-location provided opportunities for everyone to understand the different working conditions at each location:

“We really prefer to co-locate, not just the teams, but customers and other stakeholders. The first co-location involved the team members [from India] working at the customer’s site. Then, the second time was the team [members] from [the] USA, the customers, and all the stakeholders came down here [to India] and worked from the same location for a month.” —P18, Business Analyst.

Team members were comfortable getting in touch with one another for casual conversations (S 5.1.3) and contacting the customers directly for clarifications (7.1.3), mainly because of the strong relationships developed between them during the temporary co-location:
“When we started, we moved everyone to the client site, [and] worked co-located for a short period of time. When we moved back to our site, there was a very natural bonding within the team. So, once we have built that rapport, then we can talk directly with the clients or team members who are located there without hesitation.” —P35, Agile Coach.

Temporary co-location fostered cohesion within distributed teams (s 4.1). When team members returned back to respective locations after a temporary co-location, the bonding that have been developed during the co-location continue to foster informal communication between them (s 5.1.2).

6.2 Discussion

Several studies reported the impact of spatial distance on global software development [8, 26, 49, 55, 126, 133, 149, 181]. In this section, we discuss our findings in the light of related literature.

Owing to spatial distance, Holmstrom et al. [150] reported that members of a distributed team face difficulties in establishing a sense of belonging to the team. When social communication happens through communication technologies, such as e-mail, instant messaging, and the phone, team members had a feeling of being two different teams. Herbsleb and Mockus [126] reported that there exists a substantial evidence that local members of a distributed team feel less “teamness” with the remote members of the team. Spatial distance affected the usual stages where a team grows to become a coherent team with solid relationships between the team members. Without personal connection with one another, it took a while before team members felt comfortable “chatting” with instant messaging tools or engaging in casual conversations. In our study, we found that team cohesion is important for the team members to work together effectively and deliver business values successfully to customers. In order to
foster interpersonal relationships within the team, the team ambassadors travelled to different sites of the team and spent time with the remote team members. During the visit to remote sites, team members engaged in social communication (s 8.1.1) in order to build a solid relationship with one another.

Spatial distance causes coordination and control challenges because managers can no longer perform “management-by-walking” [181] and team members can no longer coordinate by “peeking over cubicle wall” [244]. Begel and Nagappan [26] discuss how to improve coordination amongst distributed teams with spatial distance. They reported team members improved coordination by scheduling visits to the remote sites of the team. The purpose of the visits were to improve understanding of work practices, priorities, and local work environment. For several team members, the cross-site visits served to “put faces to the names” and to get to know one another. Upon returning from the visits, team members were no longer hesitant to ask for help from the remote members, and they often engaged in casual conversation to resolve misunderstandings. From our study, we found teams kicked-off Agile projects with inception workshops. Team members had the initial opportunity to meet and get to know one another during the inception workshops. While only the key members of the development team attended the inception workshops, the rest of the team members met one another in person during the cross-site visits or temporary co-location. While working in close proximity with one another, the team members built rapport and relationships with each other as a basis for effective communication.

According to Eckstein [79], it is difficult to start a project if the members of a distributed team have not met in person. She suggest bringing the whole team together at one location at the beginning of the project and then rotating them from time to time during the different phases of
the project for cultivating ownership of all project essentials, establishing collective understanding and developing mutual respect among all sites. Paasivaara and Lassenue [231] suggest distributed teams arrange visits for team members frequently enough. In particular, the initial visit should be for a long stay to facilitate working together as co-located team on the project. Team members thus learn to know each other and develop joint working habits by working together at least for a short period of time. Our findings confirm the importance of bringing the whole team together at the beginning of the project. We found that the temporary co-location of a distributed team provides opportunities for team members to get to know one another and improve social communication, which in turn develops trust between them. In our study, teams rotate the location for temporary co-location in order to understand the working conditions and constraints on different sites.

Hildenbrand et al. [133] reported spatial distance exacerbates communication and coordination challenges. To overcome these challenges, distributed teams practiced cross-site visits with frequent exchange of ambassadors. They found that ambassadors enormously improved communication at both sites by facilitating the understanding of business context and cultural differences. Fowler [95] explains that an important part of the ambassador’s job is to communicate “gossip” so that formal meetings serve to discuss the critical concerns which affects the projects. When ambassadors change after a short duration, a remote team can get to know different ambassadors over the course of a project. Fowler [95] suggests “seeding” and “maintenance” visits in order to build and maintain relationships within the distributed teams. Seeding visits are scheduled early in the project in order to develop cohesion within the team, and maintenance visits are scheduled later during the project in order to maintain relationships or promote knowledge sharing.
Ehrlich and Chang [81] carried out a study to investigate how global software teams leverage expertise across spatial distance. They reported knowledge sharing is easier to occur within co-located teams than distributed teams. Members of the distributed team hesitate to seek help from one another because they could not establish solid relationships within the team. Hence, knowledge sharing is happening only between team members who are physically close or who have built a personal connection with one another. A key tenet of Agile software development is knowledge sharing within the team [51]. Co-location promotes sharing of tacit knowledge amongst the development teams, and onsite customers provide instant feedback to the development teams [23]. From our study, we found that teams practiced temporary co-location in order to get a shared understanding of the project and to facilitate knowledge sharing within the entire team.
Chapter 7

Bridging Socio-Cultural Distance

In this chapter, we present the Same Culture and Same Practices strategies for bridging socio-cultural distance.

7.1 Same Culture

Same Culture strategy aims to promote cross-cultural communication in order to bridge cultural differences.

Culture plays a major role in distributed software development because teams comprise individuals from countries with different cultures. Cultural differences between members of a distributed team, however, could lead to miscommunication and cause misunderstandings between them:

“One of the experiences is when asked a question, the answer is always ‘Yes!’ We later came to understand that ‘Yes’ means ‘Yes, I heard you’, not ‘Yes, I am going do it’ or ‘Yes, I agree to it’.” —P1, Developer.
Team members from different cultures often faced difficulties to understand hand gesture and body language because non-verbal communication is highly dependent on cultures. While communication is important for distributed teams, the national culture is often at the root of miscommunication in distributed teams:

“...body language is highly tied to culture. The head wobble can often mean ‘I don’t know’, but it can mean ‘Yes!’ and it can also mean ‘No!’ If we do not understand the cultures of team members in a different location, it can be hard to interact with them.” —P22, Scrum Master.

With national cultures varying widely across different continents, team members made earnest efforts in order to bridge cultural differences within a globally distributed team:

“...customers [from the USA] whom were working with the team members from Brazil found the cultures between them was quite different. So, we made conscious efforts to bridge the cultural gaps in order to work along with the customer as one team.” —P27, Scrum Master.

While national cultures of the team members imposed communication challenges, a distributed team ideally should adopt the organisational culture of the team in order to promote cross-cultural communication between team members:

“...we should cultivate a common [organisational] culture within a team [in order] to promote [cross-cultural] communication between team members with different national culture.” —P33, Agile Coach.

When a distributed team adopted the organisational culture, cohesion developed among team members, which in turn improved cross-cultural communication between them:
“Basically, we need a good culture within the team [in order] to feel that everyone is in the same team. Though everyone came from a different place and everyone has a different culture, different experience, and different exposure, everyone is important for the team.” —P30, Agile Coach.

In the next subsections, we describe how the Same Culture strategy promotes the cross-cultural communication between members of a distributed team. This strategy provides tactics for bridging cultural differences from the national and organisational cultural perspectives. Figure 7.1 illustrates the tactics for the Same Culture strategy.

![Figure 7.1: Tactics for the Same Culture strategy](image)

### 7.1.1 Cultural Awareness

When cultural differences exist within globally distributed teams, the team members should develop cultural awareness in order to know, understand and accept the cultures of other members in the team. Cultural awareness is important for distributed teams because culture significantly shapes the pattern of behaviour of the team members – in particular the way they communicate and interact with one another.
When team members neither knew nor understood the national culture of other members in the team, misunderstanding and unnecessary conflicts may rise between the team members.

“The Indian [team members] don’t say ‘No’ to anything. [In] the [Indian] culture, you should always obey the seniors and elders, [and] never decline a request from them. That’s one of the major problems faced by all western customers.” —P33, Agile Coach.

Even within a common language, words can carry different meanings and create misunderstandings and conflicts among team members:

“Here [in the USA] ‘as soon as possible’ means ‘rapid, fast, and you need to do it immediately’, [but] in Bulgaria, it means ‘whenever you can’. So, these cultural [differences] create friction between team members.” —P53, Agile Coach.

When working in a multi-cultural team, the team members educated one another on the cultural differences between them in order to develop cultural awareness within the team:

“...they keep nodding their heads during [verbal] communication. Many times I have whispered to them, ‘Don’t nod! They can’t understand that [body language], so you have to talk’.” —P44, Product Owner.

Cultural awareness helped team members familiarise themselves with different cultures and avoid miscommunication in a team. When team members developed awareness of another culture, they had a better chance of assimilation or acceptance:

“...being aware of the culture of individuals from distributed teams help them to fit in better in that [distributed] team. It is important to be aware of the different cultural implications at different sites.” —P5, Agile Coach.
When teams were aware that individuals from different cultures may exhibit different traits of behaviour, the teams learnt to accept the differences and find ways to work with one another:

“In Brazil, team members are used to someone telling them what they need to do, and they tend to be very courteous when they are asking for things. That’s so different with team members from the USA. We just have to learn to work with them.” —P27, Scrum Master.

Through cultural awareness, the team members developed mutual understanding and foster team cohesion within the team:

“...we have to make sure that the entire team is aware that there are cultural differences between [team] members from these different sites. In that way, we [can] still understand each other and work together.” —P46, Agile Coach.

Cultural awareness fostered mutual respect between team members and helps them to communicate and collaborate with individuals from different cultures:

“...you have to be aware of cultural differences and respect the culture of every team member, and always consider cultural differences during communication or while working together in a distributed team.” —P18, Business Analyst.

While interacting with team members from different cultures, team members sought for further clarification to improve understanding and to avoid miscommunication amongst them. In particular, when seeking for commitments from the team members, they must understand one another clearly:

“...when we’re asking for a commitment from the team members in a different site, we make sure to get a clear understanding whether they are agreeing or not agreeing.” —P22, Scrum Master.
Even when talking, team members from different cultures often faced difficulties understanding one another. In order to improve cross-cultural communication, team members were encouraged to explain ideas or opinions rather than accepting ‘yes or no’ answers:

“...it was better not to ask ‘yes’ or ‘no’ questions, [but] instead ask open ended questions such as ‘Tell me how you interpret this requirement’. It creates opportunities for them to express themselves and make sure they have a good understanding of the requirements.” — P51, Scrum Master.

Some experienced team members offered to coach the team on cultural aspects in order to create awareness on cultural differences within the team. With the experience of working with individuals from different cultures, these team members were able to relate their experience to the current situations within the team:

“We had some individuals [in the USA] and a counterpart in India who took it upon themselves to work on the cultural aspects through coaching culture.” — P1, Developer.

When teams were negotiating projects with customers from different cultures, organisations exposed team members to those cultures in order to promote cross-cultural communication with the customers:

“We had a cultural consultant who spoke to us for half a day about how to do business in a very patriarchal culture [and] that was quite an interesting experience.” —P50, Product Owner.

Cultural awareness can help team members to understand how an individual from a particular culture may communicate and socialise with one another in a team. Cross-site visits (s 6.1.2) and rotation (s 6.1.3) provide team members with opportunities to understand further the different cultures of the team members through face-to-face social communication (s 8.1.1), and thus help to bridge the cultural differences.
7.1. SAME CULTURE

7.1.2 Language Support

Language barriers indicate the difficulties faced when individuals with different native languages attempt to communicate with each other using a common language. Conversely, the “language support” tactic describes the efforts to bridge language barriers between members of a distributed team in order to improve cross-cultural communication.

With the English language as the lingua franca in global software development, team members who were non-native English speakers faced difficulties while communicating with native English speakers. Participants reported that conversations were limited during formal meetings (s 5.1.1) and the non-native English speakers needed a long time to express ideas and opinions in English:

“Meetings were [conducted] in English, [but] English was not the main language for everyone, and because of the language [barrier], they were not speaking to everyone, or they did not speak much at all. Some [team members] took a long time to express an idea properly in English.” —P7, Agile Coach.

Communication styles exacerbated language barriers within the team because the team members from different cultures inherently speak in different styles which can be difficult for someone outside that culture to understand them. Language barriers and communication styles both caused difficulties for cross-cultural communication in a team:

“There was always a communication gap between the customer and the development team [because] the development team didn’t speak English so fluently, [and] the communication style of the customer was so difficult to understand for the [development] team.” —P52, Scrum Master.
One of the communication styles of the native English speakers is the use of idioms which are closely tied to a culture or country. While team members from different cultures or different countries may not understand those idioms, they could also be wrongly interpreted, thus leading to miscommunication in the team:

“The team had hard time understanding what was being said in conference calls because team members from the USA use a lot of American idioms during conversations.” —P51, Scrum Master.

Communication may only be effective when individuals communicated in a way that each other may understand without misinterpretation. When a customer used business jargon or the development team used technical jargon, understanding one another may be difficult:

“...the customer often used a lot of business jargon.” —P52, Scrum Master.

When there were differences in pronunciation between the native and non-native English speakers, spoken accents may create difficulties for understanding what was said, especially during conversations over the phone:

“When someone’s accent is even slightly different during an audio conference, we may not be able to understand everything that they are saying.” —P14, Developer.

Organisations, customers, and development team members worked together to improve cross-communication in the team. A useful approach was that individuals consciously spoke slowly and clearly, without using jargons or idioms, in order to avoid miscommunication within the team:

“We tell them to speak slowly and deliberately so that their English is much better and easy to understand.” —P1, Developer.
With online language translation tools available on the web, team members may use them to improve written communication:

“We often write down our ideas in our own language and use Google Translate to translate them into English. It worked quite well for writing e-mails.” —P9, Developer.

Using technologies and adopting suitable techniques may contribute to improving communication. Participants reported that team members used different communication technologies at the spur of the moment. Writing or drawing on a board and combining with video conferencing has been effective for communication when team members face language barriers:

“When we start to get into intense discussions, we often go into Skype. As I am talking, I will be writing and drawing [on a white board]. You get the web-cam to point to the white board and say, ‘Look, this is what I mean!’ They can hear, see and read them at the same time.” —P3, Scrum Master.

Many organisations gave importance to language proficiency of the team members. Organisations engaged English language teachers and soft-skill trainers to provide team members who were facing language difficulties with English language and public speaking skills:

“We are not good at speaking English. We have English teachers who come to [company name] twice a week to teach English [language] and also soft-skills such as how to communicate with customers and transfer ideas to them. This was so important to improve our language skills.” —P55, Developer.

When a team member was not sure if other members of the team have understood the information shared in a conversation, it may be a good practice to ask them for clarification rather than assuming that everyone has understood what was spoken:
“...we have learned to ask for clarification during meetings that involved everyone from both sites.” —P14, Developer.

Participants reported that the non-native English speakers were asked to send e-mails answering Scrum questions prior to the daily meetings so that the rest of the team members may understand them well during the meetings:

“Before we have the daily Scrum meetings, [the non-native English speakers] had to send an email answering basic Scrum questions – what did I do yesterday, what am I going to do today, what are my problems and impediments – so that they can have it in writing. I found that working!” —P3, Scrum Master.

Concerns arising from language barriers must be addressed as early as possible in order to improve communication and develop cohesion between team members. When cross-cultural communication improved with the “language support” tactic, regular communication (s 8.1.4) between team members further contributed to developing a successful team.

### 7.1.3 Open Communication

When the individuals involved in a project (such as the development team, management and customers), practice open communication, there can be direct and honest exchange of ideas, opinions, and objections among them. When there is open communication, team members share all information relevant to the collaboration without hesitation.

Though team members must immediately bring up the impediments faced in the project during the daily Scrum meetings, some of them were still reluctant to address the problems openly. When team members hesitated to discuss problems faced in the project with fellow members of the team, open communication may be difficult in the team:
“Some team members do not openly talk about their impediments during Scrum meetings. They think that they should not be talking about their problems in open, especially when they are new to the team.” —P45, Senior Manager.

While Agile teams worked closely with customers to get requirements and instant feedback, some team members hesitated to discuss project related matters with the customers:

“...team members were hesitant to bring things up directly with the customers.” —P20, Senior Manager.

Participants reported that culture often influence the willingness of a team member to communicate directly and openly with others in the team. Some team members were less willing to speak up when a senior member was speaking due to respect for hierarchical structures within the organisation:

“In the USA, we tend to feel open or willing to express opinions regardless of the title or position on the team, whereas in India, individuals respect hierarchical structures within organisations so much that when a senior member is speaking, others are less willing to speak up.” —P51, Scrum Master.

In the absence of open communication with customers, the team members were working on the project with different assumptions instead of asking the customers for the right information needed for the project. Without open communication with one another, team members face challenges to provide value to customer while working on the project with incomplete or incorrect information:

“...team members do not ask questions to customers even when facing a problem [but] start assuming and do things in ways that create a lot of problems.” —P30, Agile Coach.
Participants (P2, P25, P30, P32, P45) reported there were team members who restrained from expressing ideas or opinions due to fear of repercussion. While these team members needed a lot of encouragement to engage in open communication with the customers, managers must create supportive environments for them to feel comfortable to express their ideas and opinions.

“We can reduce the fear by showing them what happens when a mistake is made. We make them aware that we are not going to blame them [but] we are actually going to solve the problem together. Just be open about everything. Open communication solves a lot of problems in the team.” —P32, Senior Manager.

The main purpose of keeping an open communication within a team was to foster a culture where every team member may communicate directly with one another on a regular basis without waiting for approval from the managers:

“We want everyone to be able to talk directly to the entire team and the customer on a regular basis without asking approvals from managers.” —P2, Agile Coach.

Teams strived to keep open communication in order to improve communication and foster interaction between team members. When a culture of open communication existed within the team and the wider organisation, access to project related information may be easy and fast:

“Communication needs to be kept as open as possible, and there should not be any hierarchy so that [team members] can communicate directly with customers and management.” —P29, Scrum Master.

When a culture of open communication prevailed within a team, the team members demonstrated honesty, transparency, and openness in the communication and collaboration with one another:
“Just be honest, be open and transparent so that others in the team know all your constraints. I think open communication is what gets them to keep in touch with each other, [or] have a conversation with the sponsor.” —P11, Scrum Master.

Open communication may foster cohesion within the team when team members felt comfortable engaging in direct communication with one another:

“I have a doctor’s appointment next Friday and I’ll not be available for the demo. So, I openly tell all the team during the stand-up meeting without hesitation [because] the transparency and honest communication exist across both sites [of the team].” —P24, Agile Coach.

Another benefit of open communication was that team members did not hesitate to provide honest feedback in regard to team performance, and remained open and transparent with one another in order to recognise the strengths and weaknesses of the entire team:

“After every iteration, we review the team in general. We all get direct feedback from each other [about] how we work together and what is our opinion about team performance. Everyone shares their honest opinions and gives useful feedback so [that] we can learn from one another.” – P55, Developer.

Open communication significantly contributed to the success of the projects through improving cross-cultural communication, developing cohesion, and getting team members to remain transparent with one another:

“...keeping an honest and direct communication is the primary ground for success in distributed teams.” —P38, Business Analyst.

Open communication between development team, customers, and management improved communication during formal meetings (s 5.1.1) and encouraged social communication (s 8.1.1).
7.1.4 Shared Value

Shared values are “explicit or implicit fundamental beliefs, concepts, and principles that underlie the culture of an organisation, and which guide decisions and behaviour of its employees, management and members” [1]. An organisation adopts a shared value in order to develop a common culture among individuals in the organisation. Similarly, a distributed team adopts a shared value in order to develop a common culture among team members across the different sites of the distributed team.

Participants reported a distributed team within an organisation often adopted the shared values of the organisation in order to bring the multinational, multi-cultural members of the distributed team, together. When every team member understood the core value of an organisation was fundamentally same across the different sites, they developed the sense of belonging to the organisation in general and the team in particular.

“Our organisation has got its core values which are same across different locations. When we go across to a different site, the local culture can be different, but the core values within our organisation are absolutely sacrosanct and [the] same.” —P16, Scrum Master.

One of the shared values that builds cohesive teams was team members across different sites were treated as equal as possible within the team and the organisation. When team members realised everyone was treated equally without differentiating them as local or remote team members, cohesion developed within the team:

“Agile [adoption] needs a mindset change in the team and organisation. It’s really important for team members from offshore to feel equal to the team members from onsite. Management has to make them as equal as possible, as same as possible, and create a really good working environment for the team to enjoy working together.” —P3, Scrum Master.
Participants reported that team members from different sites of the distributed team were given similar benefits or perks during travels. Hence, “shared value” developed a sense of togetherness between team members:

“When we need to travel, we get the same benefits for flights, hotels, or travel reimbursements, whether we are from India or Australia.”
—P40, Developer.

A cohesive, distributed team consists team members who are positioned at the same level despite them working from the local or remote site. When organisations did not consider one site superior to another, a sense of togetherness developed within the team:

“All the team members should feel that we are all one whole team and we are the same level. There shouldn’t be first class citizens and second class citizens [but] we should create a feeling that there is only one team.” —P7, Agile Coach.

Shared values within a team and the wider organisation promoted togetherness and developed cohesion (s 4.1), among members of a distributed team.
CHAPTER 7. BRIDGING SOCIO-CULTURAL DISTANCE

7.2 Same Practices

Same Practices strategy aims to promote the use of common practices across different sites.

A co-located team uses a common set of work practices while carrying out software development activities. Owing to knowledge sharing through frequent face-to-face interaction, a co-located team develops a shared understanding of the practices that should be in place within the team. For example, the coding standards and code coverage for unit tests are consistent between different members of a co-located team.

A distributed team, however, faced difficulties in developing a shared understanding while working from different sites. Hence, a distributed team may use different work practices at different sites of the team:

“If we were co-located, we can expect the team to work together and adopt common practices, but that’s difficult for distributed teams because the working style of team members from different countries are just so different.” —P25, Scrum Master.

When practices were not consistent across the different sites of a distributed team, the team members faced challenges in understanding the work carried out within the team:

“...without a standard practice for coding, we struggle to understand the code [because] developers across the different sites actually write codes so differently.” —P55, Developer.

Understanding the importance of common work practices for the software development activities, the team members adopted common practices across the different sites of a distributed team:
7.2. SAME PRACTICES

“We adopt common practices for team members across the globe to follow in order to work together as one team.” —P26, Agile Coach.

Despite the different work constraints at local and remote sites, distributed teams attempted to develop common work practices across sites:

“We continually find ways to share the work practices across different sites.” —P48, Senior Manager.

The adoption of a common set of work practices across different sites of a distributed team promoted cohesion between the team members:

“We continually remind [ourselves] that there’s only one team, and that’s why we [adopted] common practices at every location.” —P24, Agile Coach.

In the next subsections, we describe how the Same Practices strategy promotes the use of common practices across different sites of a distributed team. Figure 7.2 illustrates the tactics for the Same Practices strategy.

![Diagram showing tactics for Same Practices strategy]

Figure 7.2: Tactics for the Same Practices strategy

7.2.1 Shared Standards

Within software development contexts, shared standards denote the required or agreed level of quality or attainment for a process or practice.
Shared standards for coding conventions, definitions, or code coverage enable team members across different sites to be consistent with development processes or practices.

Participants (P3, P12, P14, P17, P19, P37) discussed the importance of a shared standard for coding convention. A shared coding convention kept the code consistent and promoted code readability. While implementing XP practices such as collective code ownership and continuous integration, shared standards for coding conventions were required to allow team members understand the code written by one another:

“When we come into a particular project environment, we need to ensure that everyone across different locations follows a same [coding] convention so that the completed code make sense in a context.” — P17, Developer.

Similarly, a distributed team needed a shared standard for writing quality code. When teams adopted XP practices, such as test-driven development, the team members across different sites can write code with a consistent level of quality across sites:

“...we are following the same practice for testing at both sites. We prepare test cases for unit tests and write the code to make the tests pass.” — P26, Agile Coach.

When there was a shared definition used to describe the practices or processes, team members may adopt consistent approaches for completing tasks in the software development activities. For example, there was a consistent definition for ‘Done’ among team members at different sites:

“...software developers in the Silicon Valley and Bangalore should have a common understanding for the definition of ‘Done’.” — P3, Scrum Master.
Without a shared definition within a team, approaches to carry out the tasks may vary across different sites of the team because team members developed different assumptions of what the tasks expect of them:

“The definition of ‘Done’ from developers [in India] is ‘I have written my code, and so I have finished my work’, [but] developers in [the] USA understands ‘Done’ as ‘the code is written, documentation is written, code is tested, and code is bug-free’. It is important for developers across [these] locations to get the right understanding of “Done” from the project perspective.” —P25, Scrum Master.

With code coverage giving an indication of test quality in a project, teams defined standards for the acceptable level of code coverage in order to maintain the quality of the code. Participants reported that development teams kept the code coverage at certain threshold and enforced the standards across different sites of the distributed team:

“Keeping the code coverage above a certain threshold would mean that code is above a certain quality. So, we make it mandatory for both sites that we do TDD [where] we test first and then we write the code, [and] there should be at least 90% code coverage for the application.” —P14, Developer.

Well-written unit tests maintained the quality of code during continuous integration when team members integrated the code into the wider project:

“We have metrics such as unit test coverage and code coverage within the continuous integration system that are going to give obvious warning flags if codes that are committed to the common code base no longer adhere to the defined standards.” —P47, Agile Coach.

When teams understood the importance of shared standards for a project, the team members strictly enforced rigid adherence to the standards in order to keep the practices consistent across the different sites:
“We have a shared practice for both locations in our team where we must check that whatever changes we have done has not broken any functionality anywhere. So, if we happen to break a build, then we must fix it, even if we have to stay back late.” —P12, Developer.

With shared standards, team members kept the software development practices and processes consistent across different sites, and supported Agile practices such as test-driven development and continuous integration, while collaborating on a project (s 5.1.5).

### 7.2.2 Shared Knowledge

Software development activities require a wealth of technical and business knowledge such as knowledge in programming language and knowledge of the application domain. Sharing the knowledge required for a project helped to ensure that team members had a shared understanding and same level of visibility into the project status.

“...when we have [access to] the same information, it makes sense for us to understand the project and to know what we need to do in the project.” —P37, Developer.

While a co-located team shares knowledge through face-to-face conversations, a distributed team transfers knowledge into explicit knowledge artifacts in a written, audio, or video format in order to share them with team members:

“...we focus on writing documentation to explain project information, decision choices, or process improvement because we understand the importance of shared knowledge for a team. Without a shared knowledge, we would end up moving in different directions.” —P33, Agile Coach.
Participants reported team members wrote test cases and design documents when customers request for them. Technical documents were used as a source of knowledge for complex software that were frequently revisited for upgrades:

“To some extent we explain the design choices and test cases in the code itself, [but] we do write separate documentation when the client manager or client architect request them.” —P19, Developer.

Writing documentation for codes or test cases, and updating project status for the purpose of sharing knowledge with remote members, required additional time and effort from the team members. Team members, however, took turn to write and update the information in the project repositories (s 7.2.4):

“...it is time consuming and tedious to write documents and update information in the project repository [because] the requirements and priorities change so often.” – P24, Agile Coach.

Team members leveraged information from different sources and learned from one another in order to gain new knowledge and improve skills that can be useful for software development:

“...junior developers just don’t have the experience and the exposure to understand performance impacts. Coding for performance is difficult because it comes with experience. So, a junior developer has to pair with a senior [either] from a local or remote location.” —P12, Developer.

The rapid changes in project information in Agile software development required the teams to acquire, develop, share, and store knowledge on an ongoing basis. Teams used Wikis for sharing information between sites because Wikis facilitate dissemination of consistent information across different sites. Shared knowledge can bridge the knowledge gap that arises from different levels of skills or experience between team members:
“We use knowledge repositories like Wiki to get project information or to learn the business logic for the project we are developing.” —P55, Developer.

Subject matter experts captured information to crucial and frequently asked questions in the form of written, audio, or video artifacts for storage in knowledge repositories, and reused them consistently in software development activities:

“We capture information that was asked for repeatedly and used frequently in the project. We entered technical details of every feature, even a bug, into the Wiki so that team members can search for the features, review the documents, add comments, and discuss with other members.” —P33, Agile Coach.

Capturing project information in knowledge repositories such as Wiki may be useful for team members who needed the information at a different period of time:

“We record the initial knowledge transfer sessions conducted through live web meetings, and store them together with the set of presentation slides that we prepared to answer those questions. Team members can refer to these recordings and documents that we have prepared at a later time [in order] to understand further the information.” —P47, Agile Coach.

Shared knowledge is crucial for distributed teams to have a shared understanding of the project and the progress in the project. Team members share the pain (s 7.2.4) in order to interact with each other for developing a shared knowledge (s 5.1.3).
7.2. SAME PRACTICES

7.2.3 Shared e-Workspace

A shared e-workspace is an inter-connected environment for software development activities that integrates collaborative software and communication technology for distributed teams to collaborate on a project. Sharing the project artifacts and technologies with team members allows them to communicate and collaborate despite working from different sites:

Team members shared user stories, tasks and project information across different sites of a distributed team using various collaborative software tools. Though some teams used commercial tools, there were also several effective open-source tools for them to use:

“We use an online digital task board and the open source file sharing and collaboration tools [in order] to share the requirements for the [user] stories with team members from five different locations, in four [different] countries.” —P24, Agile Coach.

A shared e-workspace provided a team with consistent, up-to-date project information across sites. Using the shared e-workspace, changes to the requirements of a project may be captured immediately:

“We work on a common platform across different locations as one team. When the customer provides a requirement change, we get to know the same [information] on the same day. So, we can hold the task for that particular sprint and we can plan for a new task without wasting our effort and time.” —P23, Quality Analyst.

Teams typically shared a common product backlog and sprint backlog using online collaboration tools in order to know the current status and the progress of a sprint and the project:

“We use the web-based backlog management tool [in order] to pull stories from the same backlog. It gives distributed teams visibility into the overall plan for product development.” —P5, Agile Coach.
With a shared product backlog and sprint backlog, the team members coordinated the development tasks through effective communication and collaboration within the team:

“Using a web-based product backlog, we are able to pull a story and read the requirements, and when we have a technical question, we put it up and wait for someone from the USA to respond to it.” — P12, Developer.

Similarly, teams shared the code-base in order to streamline the development tasks. When teams shared a code-base, the team members may implement the XP practices such as shared code ownership or frequent refactoring of the code in order to develop extensible, low-defect code with the simplest robust design:

“...both sites put all the codes in a common code repository [in order] to share code ownership and put quality into the codes.” — P3, Scrum Master.

To summarise, a shared e-workspace allowed team members to engage with one another (s 4.1.2) while using online collaborative software to view and update user-stories and project information, to pull tasks from a common web-based backlog, and to implement Agile practices such as continuous integration and collective code ownership.

7.2.4 Shared Pain

Sharing the pain, in the context of distributed teams, is the willingness to share inconveniences and difficulties that arise from distribution of a team. The willingness to share the pain developed cohesion among team members across different sites of a distributed team (s 4.1).
7.2. SAME PRACTICES

The local and remote members of a distributed team took turns either to start work early in the morning or to finish work late in the evening in order to create time overlaps (s 5.1.4). The time overlap allowed them to participate in formal meetings (s 5.1.1) and to engage in real-time collaboration (s 5.1.5):

“It’s often a case that teams share the pain of the time zones. Team members are willing to come in early and stay a little late when we need them to do so.” —P31, Agile Coach.

Participants reported that some teams practiced rotation of stand-up meetings for every iteration (P5, P16, P46) or every month (P18, P28, P40) in order to share the pain while working across different time zones:

“We start rotating the stand-up meetings. For one month it is going to be at night [in India] and the next month it will be in the morning.” —P18, Business Analyst.

When teams organised sprint planning or retrospective meetings, the schedule for the meetings can be convenient for one site but inconvenient for another, especially when there was no or minimum time overlap between those sites. A cohesive team continually shared the pain while working together from different sites:

“We alternate sprint planning meetings between mornings and evenings for both sites. It’s fair in the sense that it is convenient to one site but inconvenient to another, and we rotate it for every sprint. We share the pain, not just the glory.” —P46, Agile Coach.

While it was important to capture shared knowledge in project Wikis (s 7.2.2), the process of writing the technical and business knowledge and updating project information may be tedious and time consuming. Participants reported that team members took turns to update the Wiki with up-to-date project information in order to share the pain across the entire team:
“...we rotate to update the information into the Wiki because we didn’t want a heavy burden on the same individual or the same site.” —P55, Developer.

Setting up the communication channels for formal meetings may required time and effort from team members. Every team member within the local and remote sites also shared the pain to set up the communication channels before the meetings:

“...the phone gets a while to set up, the video conference gets a while to set up. So, we agreed to take turns to set up the phone or video conference before the [formal] meetings.” —P1, Developer.

When the members of a distributed team shared inconveniences and difficulties that arise from distribution with one another, a sense of togetherness emerged within the team and subsequently cohesion developed among team members (s 4.1).

7.3 Discussion

Several studies assert interaction among team members is an important characteristic of high performance teams [40, 89, 83]. Interaction provides avenues for team members to state ideas and opinions, listen actively to understand the concerns of team members, and provide timely suggestions to the problems faced by team members [164, 169]. Success of Agile software development significantly depends on team interaction [170, 171, 243]. Hofstede et al. [145] argue national cultures and organisational cultures have significant impact on the interaction among individuals. Organisational cultures, which are composed of practices rather than values, can be managed by changing the practices. Hofstede et al. [145] argue common practices, not common values, keep cross-cultural individuals in an organisation, such as members of a distributed team in global software development, together.
7.3. DISCUSSION

Several studies [72, 149, 152, 154, 237, 291] discuss the impact of socio-cultural distance on global software development. With team members located at different countries, Hossain et al. [154] report that misunderstandings occur due to differences in national and organisation cultures, and inconsistent work practices between sites. Hence, cohesion is challenged for a distributed team. Sprint planning meetings were audio- or video-recorded and shared with the team members for playback in order to bridge language barriers. Regular participation in formal meetings helps a distributed team to develop and reinforce consistent work practices between sites.

When teams do not adopt consistent work practices between sites, misunderstandings often occur within the team and team members implicitly show less commitment to the team [216]. Cross-cultural communication is difficult when there are differences in work styles between sites [291]. Team members ideally should break down the disparities in work styles and focus on shared knowledge within the team. With an integrated global code base and shared workspace, team members can develop a shared understanding of the project, and collaborate with one another in order to be successful in the project [237, 291].

Lee and Yong [270] reported that distributed teams use Wikis to manage project information. The teams continually wrote and updated the project information in the Wikis in order to share user stories, sprint backlogs, or project status between sites. While team cohesion was considered important for effective communication, teams also keep photos and basic autobiography of the team members in the Wikis. From our study, we found cohesion among team members is crucial for distributed teams (4.1). A cohesive, distributed team shared project-related information in order to develop a shared understanding within the team. With shared technical or business knowledge available at different sites, the team improved the ability of team members to complete the tasks successfully.
Co-located teams reduce focus on comprehensive documentation because team members share knowledge with one another through frequent face-to-face communication. Distributed teams, however, write and maintain documentation for the purpose of sharing knowledge between sites [152]. Distributed teams also emphasise written communication in order to provide non-native English speakers with written documents when facing language barriers during formal meetings [149]. From our study, we found writing technical or business description in the Wiki promoted knowledge sharing within the team, while writing e-mails answering Scrum questions helped to bridge language barriers within the team.

Besides pointing out documentation is important for distributed teams, Phalnikar et al. [237] reported subtle differences of terminology used between sites created obstacles for an otherwise well-executed project. While language barriers and communication styles are concerns for the teams, the interpretation of spoken words varies based on cultural assumptions [152, 237]. Phalnikar et al [237] suggest spreading cultural awareness helps team members to realise the need for a clear and verified communication within the team. In a recent survey [254], Forbes Insights reported language barriers have a broad and pervasive impact on business operations. Hence, two in three (68%) executives from multi-national companies required cultural awareness training in order to communicate effectively with customers and team members.

Moe and Šmite [216] investigated key factors causing lack of trust in global software teams. They reported there was poor socialisation within the team due to language barriers, which in turn, affected communication within the team. Dullemond et al. [72] reported team members from different sites find it hard to have faith in the good intentions of one another. Without cohesion, team members could not perceive themselves as part of the same team due to the socio-cultural distance. Similarly, we found team cohesion fosters social communication within a team (s 8.1.1).
Chapter 8

Impact Factors

This chapter describes how trust and senior management support influence the adoption of the Same Team, Same Time, Same Space, Same Culture, and Same Practices strategies. We first present how members of a distributed team build trust with one another and discuss how trust impacts the adoption of the strategies. Next, we present the importance of senior management support for distributed team, in terms of organisational culture, human resource management, financial sponsorship, technical infrastructure, and customer liaison. We finally discuss these impact factors in the light of related literature.

8.1 Trust

Trust between members of a distributed team is essential for them to work together as one team. A team member is deemed trustworthy when the individual is willing to respond appropriately and use existing skills and knowledge in order to benefit the team and achieve project goals (s 2.3.2). In this section, we describe how team members build trust with one another and how trust influences the adoption of the strategies.
8.1.1 Social Communication

Social communication is a form of casual communication for the purpose of socialising or getting to know one another from the professional and personal backgrounds. Many participants discussed the importance of social communication for building trust:

“Social communication is so important in Agile [projects], and it goes a long way in building trust with the team members.” —P20, Senior Manager.

Team members from different cultural backgrounds often faced difficulties to understand communication styles and work practices which are tied to each other’s cultures. Through social communication team members developed cultural awareness and built trust with one another:

“The project manager in the USA. She had great difficulty initially to understand the way we do things. She was not comfortable with us and did not much trust us until she came down to India and socialised with the team.” —P45, Senior Manager.

Participants reported that without frequent face-to-face interaction, distributed teams faced difficulties to build trust and succeed as a team:

“I think it is really hard to maintain trust, especially harder to establish trust without having face-to-face interactions. If trust is not there, it is hard to succeed as a team.” —P4, Agile Coach.

Team members developed a solid relationships based on trust with the individuals whom they are working with, especially the customers. Opportunities for building trust within the team may arise when team members initially meet during project inception workshops. A team relationship based on trust may develop a strong bonding that is the basis for effective communication while working from remote sites:
There has to be a good professional relationship with customers. They need to trust that you are going to work in a professional manner and deliver value to them. Trust will lead to effective communication when you work with distributed teams. If [team members] don’t trust you, distribution doesn’t work [because] communication will be affected.” —P11, Scrum Master.

Interaction between customers and project team members over phone or in person developed mutual respect and understanding between them, and developed long term professional relationships. Participants reported that solid relationships among team members has been helpful while making requests from customers and senior managers because they understood and trusted the team:

“We had to negotiate with the clients for some changes in a feature [but] it was not a challenging thing at all. Clients well understood why we requested for those [changes] because we know each other personally, we have spoken over the phone, and we have actually met them and had dinner together. It’s important for team members to have social interaction with product owners and key stakeholders so that they trust the team, and negotiation becomes very easy.” —P34, Agile Coach.

Understanding the importance of socialising, teams allocate time for social communication before formal meetings (s 5.1.2). Some individuals engaged in social conversation (s 5.1.3) whenever possible. Cross-site visits (s 6.1.2), however, provided opportunities for social communication through face-to-face interaction.
8.1.2 Evidence of Expertise

Teams organised verbal presentations to share technical expertise for knowledge sharing, which in turn, built trust within the teams. Preparation for the presentations as well as doing the presentation required substantive effort from a team member. The effort and the willingness to provide information and share knowledge developed trust towards the presenters:

“We have a 15 minutes presentation once in a week so that team members are able to share their own technical expertise with the team. ...all these presentations were collected and put in a shared location for the team members to view and understand them. This [activity] really helped them to develop trust on the presenter for sharing technical expertise with the team.” – P34, Agile Coach.

When team members participated in knowledge sharing activities, they gained trust from others whom benefit from those activities. There were no hesitation on what to ask or whom to ask from because they trust that no one would not look down on them for asking questions that may seem trivial to them. Trust developed naturally within a team when team members were willing share expertise and experience with the whole team:

“They were willing to participate in the knowledge sharing activities. They now realise that others will not belittle them for asking a question that may seem trivial to others. Knowledge sharing was happening without much stress on them.” —P41, Agile Coach.

When teams established and continually sustained trust, the communication across sites and collaboration between team members improved because team members were willing to share the knowledge required for the tasks and work together while supporting one another:
“We introduced bi-weekly technical huddle sessions to share technical knowledge on what we are doing and what are the things blocking our work. After every session, the communication certainly went up a fair bit, collaboration with client improved, and we understood better what we were developing. The main reason for these improvement is the trust we develop during these sessions.” – P38, Business Analyst.

Team members often travelled across sites for the purpose of knowledge sharing and getting to know one another (s 6.1.2). When team members engaged in discussion with the subject matter experts to acquire business knowledge and technical knowledge, face-to-face interaction during knowledge sharing activities fostered trust to develop between them:

“Often somebody has got some knowledge that need to be shared, or somebody wants to learn some technical knowledge available in one location, or somebody wants to learn business knowledge from customer-representatives. When team members travels to the [remote] sites, trust develops between them during discussion with subject matter experts.” – P27, Scrum Master.

With trust developed between team members through sharing expertise, the teams improved communication and collaboration with one another, and increased shared knowledge (s 7.2.2) within the team.

8.1.3 Collaboration Initiatives

When trust existed within the team, the team members implicitly or explicitly expressed interest to collaborate with one another. Trust between team members has been an important reason for effective communication within the team.
Team members kept frequent communication with one another in order to get a clear vision of project status and to understand the tasks that have been assigned to them. Collaborative initiatives allowed them to work together effectively and build a solid team relationship based on trust:

“In everything there was trust. We knew that certain tasks have been assigned to us and we had to get them done. So, either we were helping them or they were helping us. We kept frequent communication through e-mails, Office Communicator, and telephone to make sure everyone knows what we were doing. We worked well as a team [because] we had a good trusting relationship even though we have never met.” —P54, Developer.

Customers understood that development team members may need to communicate with them for different reasons, but due to time-zone differences, communication may happen outside normal working hours for either the customers or the development team members. Collaboration between development team members and customers built trust between them, and improved communication within the team:

“...customers absolutely trust me and give me their personal cell phone numbers and said, “Call me whenever!”. Communication for distributed teams actually gets easier as we build trust with each other while working together on the project.” —P18, Business Analyst.

When team members did not engage in conversations during meetings or communicate with one another, it was difficult to build trust with them. Without effective communication between them, teams members had difficulties to understand what is going on in the team. Some team members travelled to remote site to improve communication and built trust with one another:
“During retrospective meetings, some people talk [but] most people don’t talk, especially when something is going wrong. We have to figure out different ways in which we can get to understand what’s going on. So, I often have to travel between both sites and sit down with them and keep frequent communication. It’s for building that trust so that we can improve communication when we get distributed.”

—P44, Product Owner.

While knowledge sharing is crucial for the success of a project, some team members were not willing to share knowledge, ideas, or opinions with respect to the project. It was difficult to build trust when team members did not show passion for sharing knowledge and or learn from one another:

“Some team members lack willingness to share ideas with the rest of the organisation, lacks personal drive to learn, [and] their passion and curiosity is not high enough.” —P20, Agile Coach.

When trust existed in the team, however, there was no limit for information sharing with team members. The team members willingly shared all sorts of information and knowledge so that everyone may work together to contribute for the progress of the project:

“ We took efforts to ensure that all our technical decisions we communicated upfront to team members, and we explain to them in great detail why we had taken a particular decision. Therefore, it would make more sense for them to start delivering functionality.” —P38, Business Analyst.

With short sprint cycles, team members worked together closely to achieve the sprint goals. When team members delayed the tasks assigned to them, it may be visible during daily meetings. While building trust may be difficult, the excuses for not doing project tasks may easily break trust between them:
“...some team members have the tendency to delay the work [because] they are not focused on the [project] goals. When others come to realise [that] the excuses are not valid, they break the trust.” —P30, Agile Coach.

When team members often did not attend formal meetings on time, the lack of respect and collaboration caused trust to decline with them:

“...even some team members come really late to the stand-up meetings. That’s why it was difficult to build trust within the team.” —P39, Agile Coach.

Collaboration initiatives shown within the team across sites, allowed team members to trust one another. Teams often built trust while working together at real-time (s 5.1.5).

### 8.1.4 Regular Communication

When team members keep frequent communication during daily meetings or retrospective meetings, they got to know one another well and developed a sense of mutual understanding towards each other’s culture. Participants reported that promoting verbal conversation between team members may provide them with opportunities to build trust between them:

“I try to get people to know each other, facilitate meetings, and make everyone speak up during the meetings. If you can get them to talk [to each other], you can build trust and solve a lot of problems.” —P53, Agile Coach.

Customers considered team members from all the different locations to be part of one single team. Hence, customers engaged in direct interaction, preferably face-to-face interaction when possible, with team members in order to build trust between them:
“We always have someone from client side [working] with us here so that we can talk to them anytime, especially to pick up information from us, take our queries, and go back to their counterparts there to get them resolved. The customers should talk face-to-face with us often. Only then there’ll be more trust.” —P30, Agile Coach.

Regular communications allowed team members to share knowledge and project information with one another. Through frequent discussion on project requirements, trust developed between customer and the development teams:

“We have discussion with the customers on a daily basis, and involved the customers throughout the development phases. So, the [development] teams get the requirements and enough explanation from them on what we need to develop.” —P24, Agile Coach.

Teams invested in technological infrastructure in order to promote regular communication within the team. When team were aware of the availability of one another, they engaged in regular communication:

“We have invested a lot [of money] for creating smart environments [where] each other’s office is projected live onto a wall. So, we actually see the [remote] members and know what exactly the team is doing right now. We can talk to them whenever we need to.” —P22, Scrum Master.

Regular communication during formal meetings (s 5.1.1) and casual conversation (s 5.1.3) allowed team members to build and sustain trust within the team.
8.1.5 Substantive Feedback

When team members provided substantive feedback to the team, the appreciation for putting in time and effort to give the feedback developed trust in them. During daily meetings, team members attempted to provide feedback to the team with respect to the tasks carried out for the projects:

“During daily stand-up meetings, we’re interacting with the entire team. Everybody knows what we have done, everybody can take a look at the code that we have written, and everyone knows what we are delivering. There’s honest feedback going back and forth from everybody.” —P14, Developer.

Teams conducted retrospective meetings after each iteration to assess the level of success of the iteration against the iteration and projects goals. When the team members gave substantive feedbacks, the knowledge acquired from the retrospective meetings may be useful for improving the processes and practices that were in place within the team:

“...based on the feedback from clients and team members, we’ll be taking corrective actions and analysing whether the project is going in the right way and whether it can be completed in time.” – P23, Quality Analyst.

Team members provided feedback in regard to the overall performance of the team, and remained open and transparent with one another in order to recognise the strengths and weaknesses of the entire team. With substantive feedback from one another, team members sensed a feeling of trust developing within the teams:

“...we can learn from one another [and] we get feedback [about] how we work together and how is our team performance. Everyone shares their opinions and gives useful feedback. It develops the feeling of trust within the team.” – P55, Developer.
8.1. TRUST

While giving feedback was useful for the team member to improve on the processes and practices, the team members also used the feedback to reflect on their behaviour for self-improvement in order to develop a cohesive team (s 4.1).

8.1.6 Team above Self

During retrospective meetings, team members reflected on how to improve practices or processes in place within the team. When team members suggested good ideas or opinions for improvements, it was important that other members of the team consider to accept the suggestions for the benefit of the whole team rather than disagreeing for personal reasons. Team members developed trust on one another when they gave importance to the ideas and opinions of others in the team:

“We may have our own perception of how to adopt certain practices, but we also accept ideas and opinions from other members and accept them for the best interest of the team.” — P40, Developer.

When changes were suggested to a team, the team members often engaged in discussions in order to understand the reasons for adopting the changes. During the discussions, team members considered the benefits for the whole team rather than for individuals:

“When we wanted to implement some new ideas or new practices, we always discuss them with the whole team. Though bringing new changes to the team will be difficult, [but] the team members always accept the changes for the benefit of the team.” — P33, Agile Coach.

Even though some team members may not agree to decisions adopted within the team, they were still willing to accept the decisions and adopt the changes that comes with the decision. Participants reported trust developed within a team when team members understood that team’s objectives supersede those of the individuals:
“We can work together [although] we are argumentative and opinionated, and we question each other on our decisions. Finally, what matters is the team, not the self. It all comes out from the trust that we build on one another.” —P16, Scrum Master.

Team members often travelled to different sites of the team for various reasons (s 6.1.2). Due to a limited budget for travelling, the team members cautiously spent the funds so that it could benefit more team members:

“When we travel, we get the cheapest possible flight and stay in an economical hotel so that travel fund can be useful for many team members than just for the few.” —P38, Business Analyst.

When team members faced challenges in completing tasks, the team members often helped one another in order to achieve the desired goals of the project. Trust developed within a cohesive team that worked together while providing help to one another in completing the tasks for a project:

“...I knew that I wasn’t alone in doing a task. The rest of them always helped me in so many ways to get the tasks completed. Every task was [completed with] the team’s effort, not simply a single member’s effort.” —P12, Developer.

Trust developed within a cohesive team (s 4.1) where team members give importance to the whole team rather than themselves with respect to the software development activities for the successful completion of the projects.

8.1.7 Communication with Concern

When team members across sites attempted to understand the challenges and difficulties faced while carrying out the software development activities, trust may develop within the team. Communication with compassion and concern for the team members helped them to work closely in order to support one another to complete the tasks:
“...when we sincerely show care and concern for the team members, there will be natural trust on one another within the team [and] it gives them a good reason to work together in every task.” —P31, Agile Coach.

Even when a team member made mistakes while doing the work, the team must accept the mistake and help him or her to learn from the mistake rather than criticising that team member. The perceptions of interpersonal concern and mutual understanding helped to develop trust within the team:

“The idea we impart is an individual can make mistakes [but] every time if it is a new mistake, it is acceptable. So, it’s [meant] to make them realise their situation without hurting them or arguing with them.” —P33, Agile Coach.

Participants reported that trust develops when team members demonstrate concern for one another. While the constraints across sites were different, team members who took extra efforts to understand those constraints were able to communicate with team members with concern:

“...knowing the constraints for the team [in India], gives me empathy and sympathy for the [individuals] whom work under such conditions. It allows me to have an informed conversation with them when we have discussions with them.” —P22, Scrum Master.

Team members often engaged in casual communication (s 5.1.3) in order to talk to one another, not just for project related matters but also for personal reasons. Showing concerns during communication can develop trust within the teams.
8.2 Senior Management Support

Several participants point out that integrating Agile methods in distributed software development requires considerable effort from team members, particularly the senior management. Hence, distributed teams sought support from senior management and the senior management provided the support to distributed teams in order for the team members to work together as one team. Next, we describe the importance of senior management support for distributed teams in terms of organisational culture, human resource management, financial support, technical infrastructure, and customer liaison.

8.2.1 Organisational Culture

Organisational culture has a significant impact on the social nature of Agile teams in the organisations. Though team members across different countries have different national cultures, the team members working together on a project strive to develop a common culture within the team based on the culture of its organisation (s 7.1).

Senior management typically requires detailed plans and schedules, and focuses on risks and opportunities in projects, in order to commit for supporting the teams. In contrast, Agile teams focus on embracing changes rather than following plans for the development of a project. Success of Agile adoption in an organisation relies on the synergy of Agile teams and senior management:

“There is a need for change in mindset of [senior] management, towards leadership behaviours that encourage collaborative working. It needs a genuine commitment from senior management at the organisation to be a successful [Agile] team.” – P6, Developer.
Agile adoption in an organisation required support from senior management particularly in the form of acceptance and recognition to the changes that come with combining Agile methods with software development activities. Senior management recognised changes in the structure and roles of the self-organising, cross-functional Agile teams, and provided support to the team to achieve a collective success in software development projects:

“...you have to start gaining acceptance at the management level, and then from team members, so that everyone feels that they have a stake in the success of the Agile [projects].” – P44, Product Owner.

A culture established through cooperative social practices between Agile teams and senior management that understands and recognises Agile values and principles is critical for the success of Agile adoption in organisations:

“Senior management was building Agile culture within the [organisation] and transforming our organisation. Management understands the value of Agile and provides 100% support to the teams. The success in Agile [adoption] within the team is incredible. Without support and buy-in from senior management, Agile adoption will eventually fizzle out in the organisation.” – P46, Agile Coach.

Senior management in an organisation stabilises and reinforces organisational culture over time with regulations and policies for work practices and everyday activities of individuals in the organisation. Agile methods, however, may develop situations that are antithetical to typical management approaches. Since Agile methods rely substantially on interaction, collaboration, and communication, the key to successful Agile adoption in an organisation is the team. Therefore, senior management needs to create a culture that fosters Agile adoption within the team and the wider organisation.
8.2.2 Human Resource Management

Senior management must learn to manage Agile teams differently than traditional software development teams, particularly to accommodate human resource issues such as team-based versus individual rewards, skills requirements and team building activities. Ideally, senior managers must not associate team members with specific roles, and review and reward them for individual performance (s 4.1.4):

“Senior management needs to change the way Agile teams are managed. For example, senior management should consider how are individuals being reviewed and rewarded, or what programmes can be put in place to ensure that the team comes together, or what training do individuals in the team need so that the team is going to be successful.” – P46, Agile Coach.

Since interaction and communication are considered key tenets of Agile teams, senior management must give particular importance to the hiring criteria in the organisation. Individuals who do not work well together, often due to the lack of soft skills, may destroy the collaborative nature of Agile teams. Therefore, senior management was also responsible for developing team cohesion through hiring mechanisms:

“We give a lot of importance to soft skills when we hire people. We are particular about hiring people who are social and able to communicate.” – P20, Senior Manager.

Senior management may also standardise skill-set available within a team in different locations through additional training sessions, particularly on the technologies that are used in the projects. Hiring professional trainers and experienced consultants, and sending teams for training sessions and certifications may be expensive and time consuming. Therefore, senior management support was essential to design training programmes and successfully implement them across the organisation:
8.2. SENIOR MANAGEMENT SUPPORT

“We have a schedule for training programmes that everybody in the organisation needs to attend, particularly a set of soft-skills courses such as personality development, communication, management training, and leadership. And, from the technical side, we train them on software development processes, Agile practices, and what we do in the project. We put in the formal training program to make sure that everybody upgrades their knowledge and skills, and knows what the practice is.” – P45, Senior Manager.

Though individuals brought with them skills to be able to perform the job, ongoing support to upgrade existing skills or acquire new skills can be useful for the individuals. Formal training programmes that included soft-skills, technical skills, and Agile practices have been found useful to upgrade skills of team members. Managing human resource in terms of hiring right individuals to fit Agile teams, rewarding individuals for team performance, and upgrading skills of team members were important responsibilities of senior management.

8.2.3 Financial Sponsorship

Agile teams needed financial support from the organisation for software development activities, such as acquiring reliable hardware and software, purchasing information resources and attending training courses for knowledge sharing, and travelling to visit team members in different locations. Some senior managers provided continual financial support for Agile teams:

“... we also send developers for training [courses] to prepare them for [software] development projects. We have to give a lot of training before the team can start working on an enterprise-level project. The [financial] cost for these [activities] are considered high for our company [but] we still provide financial support for them.” – P29, Senior Manager.
Learning is an important aspect of successful Agile teams. Through giving continuous attention to technical skills and soft skills, Agile teams quested for different knowledge at different time from various knowledge sources. Financial support for attending conferences and training courses may be useful for encouraging team members to gain new knowledge from experts or communities of practice:

“Our company has policies for attending conferences and certification training courses. We just have to ask for financial sponsorship, and if the senior managers finds them useful and offers good value, they are willing to pay for that kind of [programmes] and invest on the employees.” – P55, Developer.

Realising the importance of face-to-face interaction and working together in close proximity, senior management ensured that whenever a new team is formed, one side of the team traveled to the other location and worked collocated for a short duration before distributing the teams. Some teams organised cross-location visits (s 6.1.2) and rotation (s 6.1.3) between team members at different sites for building rapport:

“...travelling to meet team members at different sites can be really expensive, but we provide all the support for them to meet and get to know one another.” – P20, Senior Manager.

Senior management that supports distributed teams is willing to provide financial support for encouraging continuous learning, travelling across different locations of the teams, and acquiring required hardware and software for gaining competitive advantage in Agile software development.
8.2.4 Technical Infrastructure

Co-located Agile teams communicate through face-to-face conversation and exhibit information on project status such as product backlog list, sprint backlog list, burn-down chart, or story cards using physical artifacts. Conversely, distributed teams use technology-mediated communication to carry out even basic communication activities and share project-related information between individuals in the team. Therefore, support from senior management in terms of acquiring tools for communication, collaboration, or knowledge sharing is essential for distributed teams:

“It certainly helps to have good tools that reflect team’s progress, the current [project] status and priorities, and use the knowledge [available] on the web. Tools can make it easier and effective to share information with people in remote locations.” – P20, Senior Manager.

Combination of different communication tools were particularly important for distributed teams to conserve and distribute knowledge gathered by team members in different locations. Team members found different tools were suitable for communication on different situations. Therefore, availability of several different tools may be appropriate for promoting communication and managing knowledge in distributed teams:

“We use video conferencing, and [other] tools for communication [such as] Skype, Microsoft Share View, WebEx, Messenger, and Google Documents. Some tools are free but some can be expensive but tools are good technological investment for our company.” – P55, Developer.

Participants discussed that senior management understands the importance of using technology for supporting distributed teams, and therefore often provided them with reliable technical infrastructure such as high-end computers with useful communication and collaboration tools to support team members in different locations:
CHAPTER 8. IMPACT FACTORS

“When we work from different locations, we should have good technology to support communication and collaboration.” – P28, Senior Manager.

Since distributed team members need to communicate and collaborate using technology-mediated tools, senior management should invest in suitable, reliable technical infrastructure in order to promote communication and collaboration in the team. Considerable amount of information is readily available on the Internet, and therefore additional support from senior management in the form of knowledge management tools that promote the use of information sources and integrating them into development process can be beneficial for distributed team members.

8.2.5 Customer Liaison

The Standish Group CHAOS Report [165] report that two major reasons for the 8380 projects surveyed to succeed are “user involvement” and “executive management support”. Realising the importance of customer collaboration, Agile teams advocate that customers and developers must work together on a daily basis throughout the project. Understanding the importance of customer collaboration, senior management provided active support for Agile teams in terms of customer liaison:

“...we have to work out how the team is going to work together with customers. We bring along customers to work with the team at different sites.” – P48, Senior Manager.

Though Agile methods require presence of customers or customer representatives working together with development teams, finding a customer who is willing to be involved and available for the teams across sites can be difficult. Senior management may influence customers to play a significant role in the projects through frequent interaction with development team and providing them with timely feedback:
8.3. DISCUSSION

“It has been really hard to catch up with the customer. In this case, [senior] management organise the customer to visit different sites from time to time.” – P55, Developer.

In traditional software development, customers and developers do not engage in direct communication because business analysts often draft out project requirements for developers and customers do not see the software until project completion. This may lead to miscommunication between the customers and developers. To be successful in Agile software development with distributed teams, senior management must request customers for a certain level of collaboration and time commitment from them:

“The expectations were not set properly. The customer would throw a lot of work onto the team and expect the team to deliver the work and understand the whole vision. But, the team didn’t exactly get the whole picture of the vision. So, the VP of Delivery decided that we flew the customer in with the team for a week. All the features that he wanted to deliver by that deadline, we sat and estimated but it wasn’t possible.” – P52, Scrum Master.

Besides individuals in the project teams, Agile adoption in software development also affects customers where significant level of commitments in terms of time and effort are expected from them. Close collaboration between customers and project teams can (a) ensure project requirements are clearly understood, (b) provide development team with early feedback on acceptance tests, and (c) lead to project success.

8.3 Discussion

In this section, we discuss the importance of trust and senior management support for distributed team in the light of existing literature.
8.3.1 Trust

Trust plays an important role in determining the success or failure of software development projects for Agile teams. The knowledge on what causes trust to decline, what are the adverse effects of absence of trust and how to build trust in a team, can create an awareness on the importance of trust in software development teams. Several studies investigate the importance of trust in software development teams [17, 163, 179, 204, 216, 228, 256].

On investigating mature XP teams, Robinson and Sharp [251] argue absence of trust would affect the sense of respect, responsibility, concern for the quality of working life and faith in the ability of the Agile team, and members of the team would doubt that the team as a whole could deliver business values to customers. We found that, in the absence of trust, team members exhibited lack of commitment to the work being carried out. Team members were not engaged in useful discussions, or they were not effectively participating in the meetings, or they were not prepared for sprint review meetings with the customers. The team members disregarded their responsibilities to the customer and fellow members of the team in one way or another.

Based on a study on customer communication development, Korkala, Pikkarainen and Conboy [171] found the lack of trust was one of the reasons for the customer not involved in the implementation of distributed Agile projects. The researchers suggest efficient communication is one of the most essential factors in distributed software development. Piccoli and Ives [238] reported the findings from a longitudinal study of virtual teams that incongruence, particularly obstacles to effective communication, and reneging create the potential for trust decline. Similarly, we found ineffective communication such as during the daily meetings and sprint review, and failure to recognise the need to use different communication tools, created circumstances for trust decline.
8.3. DISCUSSION

Moe and Smite [216] conducted a study to understand the causes and consequences of lacking trust in global software development in four software projects. All these projects report that lack of trust affected team performance and resulted in a decrease in product quality and team velocity. In this study, we found that, in the absence of trust, team members were less focused on the project goals, the projects were behind schedule, and the project deliverables did not meet customer’s expectation.

Through a case study of an outsourced information systems development project, Lander, Purvis, McCray and Leigh [179] argue building trust amongst team members in different locations is especially difficult because the individuals involved in the project often have little or no prior experiences working together with other individuals in the team, and yet rely on one another’s expertise and judgement for a successful project. Oza, Hall, Rainer and Grey [228] conducted a study based on an empirical investigation of eighteen mature software companies located in India. The researchers describe several critical success factors to achieving an initial trust, and eventually maintaining trust in software outsource relationships, and suggest that trust is considered to be very fragile in outsourcing relationships. Our participants acknowledged building trust between team members from different locations in project-oriented contexts can be difficult, and therefore some distributed teams take the necessary measures to avoid the causes for trust to decline in the team.

On investigating trust in virtual teams, Hung, Dennis and Robert [157] propose an integrated model of trust that encompasses both the traditional view of trust and the swift trust found in virtual teams. They argue computer mediated communication environments have indirect influences on trust and trust-behaviours within virtual teams due to higher perceived risk or the likelihood of disappointing outcomes. Therefore, virtual teams develop trust through the assessment of the other individual’s
ability, benevolence and integrity rather than through third party information, social categories, roles and rules. They recognise that computer mediated communication, mainly with fewer opportunities to engage in traditional, face-to-face trust building activities, presents significant challenges for virtual teams to develop trust.

While trust among team members is imperative for software development projects, substantial effort are required to build trust within a distributed team [246]. McHugh et al. [204] suggest Scrum ceremonies increase trust in the team by providing transparency and visibility of project status, enhancing accountability and collective responsibilities, increasing open and frequent communication, and sharing of knowledge and obtaining feedback. Thus, adopting Scrum and following the Scrum ceremonies provided members of a team ways to build trust with one another.

Distributed teams in Agile software development make conscious efforts to improve team interaction and build strong working relationships among team members to allow trust to build naturally in the team [319]. Regular face-to-face meetings and effective communication are important for building trust [303]. Team members appreciate the culture training initiatives that allow the customers to trust the team for a long term working relationship [17]. Frequent visits by distributed partners and sponsor visits, and building a cohesive team culture have been contributing significantly for building trust in distributed teams [246]. Our study confirms social communication during cross-site visits particularly has been useful for building trust between team members. However, we also found exposing expertise of team members, particularly through knowledge sharing activities, has been effective to build trust across different sites. While circumstances across sites can be different, we found trust develops when team members demonstrate concern and empathy for one another.
8.3. DISCUSSION

8.3.2 Senior Management Support

The importance of senior management support for co-located teams [47, 74] and distributed teams [84, 246] in Agile software development has been widely acknowledged. Agile teams flourish with continual support from senior management at their own organisations [139].

Ramesh et al. [246] investigate challenges that arise from blending agility with distributed development and how those challenges can be addressed. They studied three organisations that adopted Agile methods in distributed software development, and identified several practices to address challenges to communication, control, and trust across distributed teams. They reported that significant support from senior management in terms of improving communication, facilitating knowledge sharing, and building trust, demonstrates a balance between Agile adoption and distributed software development.

Livermore [193] reported senior management support for software development teams has a strong impact on Agile adoption in organisations. He found a significant correlation between the success of Agile adoption and senior management support. He suggests senior management should ideally provide Agile teams with support in terms of access to useful resources and training on software development methodologies.

Through an experience report, Farmer [84] reflects on the success of Agile adoption in a large, distributed team. She reported one of the major factors that contributed to the success of Agile adoption was the timely support from senior management, especially in the area of customer liaison and autonomy in decision making within the team.
Rottier and Rodrigues [253] discuss experience of the software development team in a medical device company. They reported substantial support from senior management has led to the success of Agile adoption within the software development team. Senior management encouraged other individuals in the company to attend the daily Scrum meetings and sprint review meetings, and developed a shared understanding of Agile software development within the company.
Chapter 9

Agility in Context

In this thesis, we present the strategies which a distributed, Agile team adopts in order to become one team. We, however, do not prescribe a distributed team must adopt every strategy but rather selectively adopt a strategy or a combination of strategies which are contextually relevant to the team. We support a contextual approach to adopting the theory of one team, where a distributed team adopts only the strategies which suit the project context.

While there are several definitions for context (for example, Dey [65] and Chen et al. [44]), we found Henricksen and Indulska [124] provide a concise definition:

“The context of a project is the set of circumstances surrounding the project that are potentially of relevance to its completion.”

In this study, we found a number of circumstances which are relevant to the adoption of the strategies: time overlap, number of sites, team size, technical infrastructure, and project criticality.
9.1 Time Overlap

Due to time-zone difference between sites, most participants reported that there was little time overlap for effective synchronous communication among the team members. Though teams often stretch or shift the normal working hours (s 5.1.4) to create enough time overlap for formal meetings (s 5.1.1) or real-time collaboration (s 5.1.5), the team members were burdened with long working hours or felt frustrated spending personal time for attending conference calls:

“The team members [in India] were really frustrated because we were spending a lot of personal time for communication with [team members] in the USA.” —P19, Developer.

When there was no time overlap between sites during normal working hours, the teams were not able to organise joint formal meetings across the sites. While joint meetings develop shared understanding among members of a distributed team located across different sites (s 4.1.3), the distributed team faced much difficulties to find a suitable time for the team members to be available for formal meetings:

“We tried to have joint meetings but due to the time zone difference, we often had separate meetings.” —P11, Scrum Master.

When team members were not sure about the availability of one another, they hesitated to engage in synchronous communication with team members from remote sites. Team members instead emphasised on asynchronous communication such as sending e-mails when it is difficult to engage in synchronous communication with team members at remote sites:

“Though there’s a need to talk to them [in India] directly, but at one point [of time] we decided to focus on written communication. It was hard because we had to call them late in the evening.” —P2, Agile Coach.
9.1. **TIME OVERLAP**

Several participants (P20, P21, P22, P27, P32, P40, P50, P55) reported the benefits of distribution across sites with adequate time overlap such as fast feedback and effective knowledge sharing. Synchronous communication was preferred when time overlap was available across sites. Team members engaged in spontaneous conversations (s 5.1.3) whenever they need to ask for information from team members at remote sites:

“...the time-zone difference was minimal. We often talk over the phone with team members at the remote site during normal working hours. There was a lot of sharing of information with the wider team.” — P27, Scrum Master.

When teams were separated across sites in the same time-zone, the team members were able to engage in synchronous communication without having to stretch or shift working hours:

“With Cape Town and London almost at the same time-zone, the distribution was ideal for communication.” — P32, Senior Manager.

With temporal co-location between sites, the team members had frequent casual communication and close collaboration within the team:

“We’re in the same time-zone with the remote site. So, we are able to have frequent conversations with them anytime of the day and work together closely as if we were co-located.” — P55, Developer.

Synchronous communication provides teams with opportunities for instant feedback and effective knowledge sharing through verbal communication during formal meetings (s 5.1.1) and casual conversations (s 5.1.3). Without adequate time overlap, however, real-time collaboration can be difficult. In order to create time overlap or increase existing time overlap, shifting work hours is preferred to stretching normal work hours (s 5.1.4).
9.2 Number of Sites

Though most teams were separated across two countries, there were teams separated across several countries or several sites across different countries. When teams were distributed across many sites, coordination for formal meetings and tasks were difficult. While joint meetings were difficult across many different sites, teams organised separate meetings and used asynchronous communication approach to update project status:

“...we [in the USA] had two daily stand-ups, one in the morning and one in the evening, [in order] to coordinate with team members in Australia and India. We had to put in a lot of efforts to communicate with one another.” —P8, Developer.

Besides the concern of time overlap across these sites, teams also needed to consider the availability of team members at each site to carry out synchronous activities:

“The team is in five different locations; two locations in India, and one each in Poland, Finland, and Sweden. It’s always a challenge to bring all these [team members] to a joint meeting.” —P24, Agile Coach.

Though team members visited one another at remote sites from time to time (s 6.1.2), travel to several countries to visit the team members can be expensive and time consuming. Team members were also not able to spend much time at each site in order to get to know one another and build a solid relationship between them. Therefore, teams preferred temporary co-location (s 6.1.5) to get the whole team together at the customer site rather than organising separate visits to those different sites for each team member:
“...the project was distributed between five cities across four countries. The project sponsor was in Baghdad, but [development] team members were in Amman, Bangalore, New York, and Boston. It has been hectic because of such a distribution.” —P11, Scrum Master.

Agile coaches preferred to travel to different sites of the team in order to build a personal connection with team members and conduct effective coaching (s 6.1.4). Travelling to many sites across different countries, however, needs substantial time, effort, and budget:

“...with teams located at four different sites, coaching them as a team is tough, and travelling to meet them and to get to know them is also tough.” —P35, Agile Coach.

With teams comprised individuals from many different national cultures, language barriers and communication styles strongly affected cross-cultural communication. Teams spread cultural awareness (s 7.1.1) and provided language support (s 7.1.2) in order to improve cross-cultural communication:

“...with team members distributed across India, Ukraine, and Australia, the different cultures of the team members often cause conflicts [within] the team. One concern was the expectation, assumption, and aspiration were different between sites.” —P33, Agile Coach.

While national cultures cause challenges for communication and collaboration across sites, teams focused to develop a culture within the organisation in order to work together effectively. For example, distributed teams practised open communication (s 7.1.3) and adopted shared value (s 7.1.4) in order to develop a positive organisation culture:

“...distributed across the USA, India, and the Netherlands, we have wide cultural differences between the team members. It is not easy to get them to change the behaviour to fit the team.” —P3, Scrum Master.
The numbers of sites for the distribution affected the ability of the teams to adopt strategies in order to produce temporal co-location (s 5.1) and physical co-location (s 6.1), and significantly increased the challenges in cross-cultural communication (s 7.1).

### 9.3 Team Size

While most teams typically started the projects with small team sizes, several teams, however, scaled up in order to accommodate the increasing complexity of the projects. When the team size was small, almost all members of a distributed team had the opportunity to travel to different sites for inception workshops or visiting team members:

"...we had four team members in the UK and three in India...throughout the course of the project, the team members from the UK went to India a couple of times." —P50, Product Owner.

In order to work closely with the customers, teams co-located at the customers sites for a period of time. With small team sizes, the teams easily adopted the practice of temporary co-location because the cost of travelling and logistics required were manageable for the projects:

"...when we have a small team, moving them across different sites, especially to the customer site, is quite easy and does not cause inconveniences for the customer." —P42, Senior Manager.

Decision to travel to remote sites for visiting one another was based on the team size at different sites. In order to minimise the cost of travelling and the inconveniences of travelling to different countries, the team members from the site with smaller size travelled to meet the rest of the team members in a remote site:
9.3. TEAM SIZE

“...we did not travel to [the] USA [because] there were 16 to 19 members at Bangalore and Hyderabad [in India] but six in Chicago. So, all the team members from [the] USA and the client travelled to India to meet us.” —P39, Agile Coach.

When a distributed team has a significantly large number of members, many team members did not get the opportunity to visit one another due to the high cost associated with the visits. While only the key members of the team travelled, the rest of them did not meet face-to-face or get to know one another in person:

“We have almost 50 team members distributed across three countries for the same project. So, once I travelled to Ukraine and spent one month with them. We don’t often get team members to travel from one location to another, mainly because of the cost for travelling.” —P33, Agile Coach.

Members from a large team often had little time to talk during formal meetings. While formal meetings consumed a lot of time for the whole team, yet the actual time available for each member to talk was not enough to get to know one another or to build solid team relationships. Without visiting team members at different sites, teams faced challenges to develop a sense of togetherness within the teams:

“...with a large team size, it can be difficult for everyone to know each other and develop good team relationships. The amount of time that we spend talking to each other is so small.” —P25, Scrum Master.

When verbal communication was limited to a short time during formal meetings, team members faced difficulties to develop a shared understanding with everyone in the team. Unless there exist an adequate time overlap, knowledge sharing was difficult because team members may not have opportunities for casual conversations outside formal meetings to share experience and knowledge with one another:
...one major concern was knowledge transfer between the large number of team members. We don’t even know who has what experience that should be shared across the team.” —P27, Scrum Master.

Team size impacted the decision for travelling to attend the inception workshops (s 6.1.1), cross-site visits (s 6.1.2), and temporary co-location (s 6.1.5). Even when team practiced travelling across sites, with a large number of team members, there was still many team members who had not travelled to a remote site for building rapport with one another. During formal meetings (s 5.1.1), team members may not have much opportunities to engage in conversation.

### 9.4 Technical Infrastructure

Distributed teams required effective infrastructure for communication and collaboration with one another across the different sites. When members of a distributed team interact over time and space, the technical infrastructure available to them determined the effectiveness of the interaction within the team. Without the availability of proper technology and tools, teams failed to complete the project:

“This project failed because we did not have the right communication technology in place. We should have used video or voice instead of only written communication. Without [collaboration] tools, we could not collaborate with them [at remote site].” —P9, Developer.

Team members used different techniques for synchronous and asynchronous communication. Some individuals, however, preferred written communication, while others preferred video or phone communication. Team member’s preferences for a specific technique restricted the overall communication within a team:
“Often someone prefers e-mails over phone calls, [but] when we are distributed, we should be able to use different technologies for communication.” —P20, Senior Manager.

When suitable communication and collaboration technology was not available, teams faced difficulties with synchronous communication and real-time collaboration. Video and audio communication provided teams with opportunities to talk to one another for acquiring immediate feedback or sharing knowledge. Without video or audio communication, there was limited synchronous communication and subsequently misunderstandings and conflicts developed between team members:

“We could have done Skype but we didn’t do it because we had some technical difficulties. We didn’t have a good infrastructure for video communication. I think video [communication] would have helped us to develop better working relationships, or we would just had fewer misunderstandings.” —P51, Scrum Master.

When organisations limited the availability of communication tools for the team, team members could not leverage tools for communication across sites:

“We need web cameras but it’s against our organisational policy. So, we don’t use web cameras in our projects.” —P24, Agile Coach.

The infrastructure for Internet within an organisation must be good enough to support synchronous communication and the use of collaboration tools. Without a good technical infrastructure, teams spent a long time to organise a formal meeting, or often experienced poor quality voice and video:

“In [this location], the Internet connection is patchy sometimes and we have power outages a couple of times a week.” —P22, Scrum Master.
“There are challenges in India [where] the technical infrastructure isn’t strong. So, the phone or video conference gets a while to be set up. [Though] we only have a 15 minutes daily Scrum, but sometimes the entire process [to set up] will take an hour.” —P1, Developer.

The availability of suitable technical infrastructure was crucial for effective communication and collaboration across sites. When distributed teams are provided with proper technical infrastructure, teams can organise effective formal meetings (s 5.1.1) and engaged in casual conversations (s 5.1.2), (s 5.1.3) when team members are available. Teams also worked together at real-time (s 5.1.5) in order to get instant feedback and share knowledge with one another.

9.5 Project Criticality

Criticality is an indication of how much is “at stake” in a project and the severity of the activities in a project, which shows the total effect that its uncertainty is having on the project [155]. Criticality is in large part dictated by the project’s goals and determined by the development team and customer. We found levels of criticality can vary – ordinary, minor, major, or severe – depending on the diverse and complex tasks within the project. Ordinary indicates that customer is comfortable with the progress of the project; minor indicates that current progress of the project may cause a loss of customer comfort; major indicates that the current progress of the project may cause loss of money but can be fixed by tactful intervention from stakeholders; severe indicates that the current progress of the project may cause loss of money and future business from customer. The project criticality may change when situations changes such as when a customer is unavailable for providing timely feedback or requirement changed to include a crucial feature:
“...decisions are made based on how critical is a project situation, whether it is ordinary, minor, or major, or at times severe. This situation changes when customers are not available and sudden changes are requested for the project.” – P28, Senior Manager.

When there were high levels of complexity and ambiguity within a project, team members gained better understanding of situations in the projects through face-to-face meetings, which was facilitated through cross-location visits:

“Right now something critical is going on, and we need some sort of complex information from the customer. Two people from offshore site went to the customer’s site to work through with all those concerns. We learnt a lot of things from them and the entire team benefits from this knowledge transfer.” – P26, Agile Coach.

Realising the benefits of co-location, some teams went so far as to move the whole development team to the client’s location for the entirety of the projects:

“...we rather do it onsite at client’s location itself. We finish up the project from the client’s site, and then come back here. So, in that way, everyone in the team can get to interact with client and understand the client’s mindset, and we’ll have a better opportunity to solve their problems.” — P36, Quality Analyst.

The evaluation of the criticality of a project may be complex and filled with uncertainty and vagueness. The criticality of a project may change during the course of the project. The decision to travel to a remote location for a short period of time (s 6.1.2) or work co-located with the customer (s 6.1.5), and the frequency for these travels, was often based on the criticality of a project.
9.6 Discussion

Dybå et al. [76] suggest context is a central concept in empirical software engineering research. They argue empirical software engineering research should be “contextualised” in order to provide an in-depth understanding of what works for whom, where, when and why. It is through the context, interpretation and evaluation that empirical software engineering research can become adequately comprehensible. According to Dybå et al. [76], the context of a research study, which depends on the research phenomena under scrutiny, must be articulated clearly, particularly on how contextual influences operate, in order to transfer evidence generated from one context to another. In this study, the project context which influence the strategies has been described in terms of the time overlap between sites, the number of sites of the distribution, size of the whole team, technical infrastructure for communication and collaboration, and criticality of the projects. Our intention to provide distributed teams with adequate description on how context impacts the adoption of the strategies may contribute to the successful uptake of research into practice.

Ambler [13] suggest tools and technologies that support distributed development, enhance collaboration between disparate team members or automate as much of the work as possible significantly contribute to the success of scaling Agile. Sutherland et al. [292] explain how distributed teams used tools for communication and collaboration in order to manage the product backlog and sprint backlog, to share project information and documentation, and to facilitate the software development process. Similarly, Hossain et al. [152] suggest distributed teams that consider to use Scrum need a wide range of tool support for communication, collaboration, project management, issue checking, bug checking or managing backlogs. They found the practice “proactive resource management” helps ensure that Agile teams have the necessary technical infrastructure
and tools to support Scrum practices in global software development. In our study, we found distributed teams in Agile software development depend extensively on technology for communication and collaboration. In particular, communication tools were used for daily Scrum and collaboration tools were used for e-collaboration on a daily basis. Access to suitable technical infrastructure thus directly influence the adoption of the strategies, such as the Same Time strategy which promotes synchronous communication and real-time collaboration across different sites. During field observations, we noticed our participants were using different tools in their projects (s 10.4). Though some team were using the free, open-source tools available online, we found many teams were using vendors’ products, mostly for reliability, support and completeness.

Agile methods in general focus on small, co-located teams, in which customers are an integral part of the teams, and emphasis on testing before coding and frequent feedback into updated requirements – a context called the Agile “sweet spot” [174, 247]. Hoda et al. [135] argue it is unsurprising that projects in the “sweet spot” benefit from the use of Agile methods such as Scrum [266] and XP [22]. In practice, many projects fall outside the Agile “sweet spot”: projects with a large number of team members, projects with distributed teams or projects with inadequate customer collaboration. Hoda et al. [135] argue development methods and practices must be adapted to fit their projects’ contexts. Understanding the context of agility is important for other Agile teams to adapt development processes to fit their projects contexts [135]. Hence, we have described in detail the context of projects investigated in this study, and in particular the circumstances that impact the adoption of the strategies.
Chapter 10

Discussion

In this chapter, we discuss several important aspects of our research study such as the nature of contribution of this thesis, the boundaries of our study, our experience from field observations, how we selected a suitable theoretical coding family, and the evaluation and validation approaches of our study.

10.1 Nature of contribution

A recent systematic literature review on agility reports that the majority (70%) of the existing literature is in the form of experience reports and that there is a paucity of empirical studies analysing the applicability of agility in distributed software development [162]. A number of studies [79, 285] point out that there is little knowledge on the overall combination of agility and distributed software development:

“... we still do not understand fully the limitations and viability of Agile methods in seemingly incompatible environments of distributed software projects” [280].
CHAPTER 10. DISCUSSION

Many researchers [121, 162, 170, 230, 279, 280, 285] suggest there is a need for in-depth, empirical investigation on combining agility and distributed software development in order to provide practitioners with useful insight for decision-making throughout the phases of software development:

“... empirical studies identifying different dimensions of the distributed software development—Agile fit are not well developed conceptually” [285].

This thesis proposes “The Theory of One Team” which explains how a distributed team in Agile software development adopts explicit strategies for bridging spatial, temporal, and socio-cultural distances, while facing critical impact factors, in order to become one team. This thesis has a number of novel contributions to knowledge as follows.

(i) There exist a number of fundamental differences between Agile software development and distributed software development, particularly in terms of communication, coordination, and control [246, 280, 290]. Agile software development is associated with face-to-face communication, change-driven, and cross-functional teams, whereas distributed software development is associated with computer-mediated communication, plan-driven, and strict role separation [246, 280, 290]. A number of studies [7, 92, 280] characterise the two approaches as opposite extremes on a continuum, thus the combination of the two approaches is a recipe for failure. In contrast, this thesis presents empirical evidence from 38 different software companies across different countries suggesting the combination of the two approaches brings significant benefits to software development teams. While separated through space, time and culture, a distributed team using Agile methods, however, adopts a particular strategy or a combination of strategies in order to work together as one team. Hence, this thesis provides actionable advice for practitioners embarking on a quest to use Agile methods in distributed software development.
10.1. NATURE OF CONTRIBUTION

(ii) Several recent studies [162, 280, 299] report there is still no deep, theoretically grounded, knowledge to understand the applicability of Agile methods to distributed software development. While practice has always been ahead of research, it is a challenge for researchers to gain a comprehensive understanding of what is going on and why practitioners do what they do while adopting agility in distributed software development. More empirical research on this area, however, would benefit the software companies that are considering using Agile methods in distributed software development [162, 170, 230].

Based on a large-scale empirical research study, this thesis explains what is happening in the area studied rather than what should, could or ought to be happening because the theory emerged from analysis of the data acquired from participants. In particular, this thesis answers the “what is happening”, “how it is happening” and “why it is happening” questions within the research phenomenon and conceptualises the relationships among the emergent constructs (s 3.8). Using the “what–how–why” pattern, this thesis explicates the theory of one team in a comprehensive, conceptual way in order to provide practitioners with in-depth, useful insight for combining agility and distributed software development.

(iii) A large number of studies [17, 204, 216, 228] reports the importance of trust for software development teams and, at the same time, points out that building trust in a distributed team can be difficult because the team members often have little or no prior experience working together. While software development process heavily relies on how individuals in a team work together, the concern for building trust is imperial for Agile teams [204, 280]. Despite the emphasis on building trust, little is known on how members of a distributed team in Agile software development build trust with one another across spatial, temporal and socio-cultural distances [171, 216, 246].
This thesis presents seven techniques for building trust in Agile software development with distributed teams (s 8.1). In particular, this thesis describes in detail how a distributed team adopts a particular technique or a combination of techniques in order to work together as one team while separated across spatial, temporal and socio-cultural distances.

(iv) A wealth of literature points out the importance of senior management support for software development teams. Senior management support, however, may not be readily available to the software teams because senior management is often reluctant to accept and recognise the changes that comes with combining agility with distributed software development [139, 246]. A large number of studies [47, 74, 84, 193, 245, 246, 253] describes how senior management support impacts the Agile adoption in organisations. This thesis, on the other hand, describes how senior management support impacts the ability of the members of a distributed team to work together as one team while separated across spatial, temporal, and socio-cultural distances. Hence, this thesis serves to inform senior management the importance of providing continual support for distributed teams from a number of aspects such as human resource management and customer liaison (s 8.2).

10.2 Relationships between strategies

In this section, we discuss the relationships between the strategies, and how a particular strategy affects other strategies which can affect a distributed team bridging the spatial, temporal, and socio-cultural distances in order to become one team.
10.2. RELATIONSHIPS BETWEEN STRATEGIES

10.2.1 Same Team strategy

Same Team strategy aims to develop team cohesion among members of a distributed team (s 4.1). Cohesion is seen as strong predictor for the performance of a team because a cohesive team works together in pursuit of its goals despite difficulties faced within the team. Hence, team cohesion is crucial for members of a distributed team to work together while separated through space, time, and culture (s 4.2).

Cohesion among team members affected synchronous communication within the team. Without cohesion, team members hesitated to participate in casual communication with one another (s 5.1.2) and to engage in real-time collaboration with the whole team (s 5.1.5). When a distributed team realised communication is important but affected due to distribution, the members of the team found ways to develop cohesion among the team members:

“...team cohesion leads to effective communication; without [cohesion], communication will be affected in a distributed team.” —P11, Scrum Master.

When there’s cohesion among members of a distributed team, temporal distance was not a matter for synchronous communication to happen in the team. Shifting the normal working hours (s 5.1.4) provided the team members with adequate time overlap for synchronous communication. Unsurprisingly, the members of a cohesive, distributed team organised and participated in formal meetings (s 5.1.1) and informal meetings (s 5.1.2), and, without hesitation, engaged in casual communication (s 5.1.3). A cohesive team also engaged in real-time collaboration (s 5.1.5) using different tools and techniques in order to improve interaction and to acquire instant feedback from team members.
Owing primarily to team cohesion, a co-located team recognised the need to use, and subsequently used, a set of common practices such as coding standards within the team. On an ongoing basis, the co-located team also re-evaluated the use of the common practices for continual improvements:

“...there’s a common set of practices used in co-located teams, [but] that’s not the case in distributed teams.” —P53, Agile Coach.

Cohesion among team members affected the use of common practices across the different sites of a distributed team (s 7.2). When there’s cohesion among team members, sharing inconveniences and difficulties that arise from distribution was not a matter for the distributed teams (s 7.2.4). A cohesive, distributed team continually shared the pain in order to support the team members in every possible way.

A cohesive, distributed team also developed shared standards so that team members at different sites of the distributed team can be consistent with development processes or practices (s 7.2.1). Despite distribution, team cohesion promoted knowledge sharing activities which in turn helped to develop a shared understanding among team members (s 7.2.2). In a cohesive, distributed team, knowledge sharing activities happened naturally without much effort from managers:

“... with cohesion developing in the team, they came forward to share their technical expertise with others through regular presentations.”
—P32, Senior Manager.

Same Team strategy primarily affected the Same Time and the Same Practices strategies. Hence, we propose a distributed team adopts the Same Team strategy to develop team cohesion in order to promote synchronous communication among members of the distributed team and to promote the use of common practices across different sites of the distributed team.
10.2.2 Same Time strategy

Same Time strategy aims to promote synchronous communication among members of a distributed team (s 5.1). When there was adequate time overlap across the different sites of a distributed team, the team members acquired almost instant feedback from one another. In contrast, when there was little or no time overlap, instant feedback was almost impossible. A distributed team adopts the Same Time strategy primarily to get instant feedback from team members:

“...even when working across extreme time zones, it is important to find ways to get instant feedback.” —P28, Senior Manager.

Synchronous communication affected the use of common practices across the different sites of a distributed team (s 7.2). Knowledge sharing was paramount for working together in Agile software development (s 7.2.2). Therefore, when synchronous communication is possible, distributed teams often organised sessions for sharing knowledge, experience, and skills with team members. During these sessions, team members interacted with subject matter experts using video or audio communication to learn, understand, or improve software development processes and practices in place within the team. When synchronous communication is challenged, however, subject matter experts spent significant amount of time and effort sharing knowledge using Wikis, in which written, audio, or video artifacts of knowledge were stored in knowledge repositories for team members to use when required:

“...subject matter experts prefer live sessions to share knowledge, [but] time zone often burden them to come up with documentation for storage in knowledge repository.” —P47, Agile Coach.
Real-time collaboration (s 5.1.5) provided a distributed team with opportunities to use shared standards such as coding convention or code coverage across different sites of the distributed team. When working together at real time, interaction among team members facilitated knowledge sharing in action:

“When we work [together] at real time, we get to learn from each other and, almost immediately, we can practice what we had just learnt.”
—P40, Developer.

Synchronous communication also affected cross-cultural communication among members of a distributed team (s 7.1). When team members interacted with one another, they realised there was cultural differences between them and subsequently realised the importance of bridging the cultural differences (s 7.1.1). Therefore, team members educated one another on the cultural aspect through continual interaction between them:

“...we learned about each other’s culture and how cultural differences affect our daily work during casual conversation with one another.”
—P44, Product Owner.

Cross-cultural communication improved in a distributed team when the team members engaged in open communication, in which a team member interacts without hesitation to share ideas, opinions or objections with the whole team (s 7.1.3). When there was adequate time overlap across different sites of a distributed team, continual interaction among team members fostered the practice of open communication within the team:

“...the regular interaction between them gradually helped them to talk to each other directly and give honest feedback on the subject that was discussed at the meeting.” —P51, Scrum Master.
10.2. RELATIONSHIPS BETWEEN STRATEGIES

Same Time strategy primarily affected the Same Practices and the Same Culture strategies, and therefore adopting the Same Time strategy is important for bridging the socio-cultural distance. Hence, we propose a distributed team adopts the Same Time strategy to promote synchronous communication in order to promote the use of common practices across different sites of the distributed team and to promote cross-cultural communication among members of the distributed team.

10.2.3 Same Space strategy

Same Space strategy aims to promote temporary physical co-location among members of a distributed team (s 6.1). Co-location promotes team cohesion through frequent face-to-face interaction among team members. In contrast, distribution adversely affects team cohesion because members of a distributed team often have less or no opportunities to interact with one another.

Co-location during an inception workshop provided team members opportunities to get to know one another in person right from the start of a project (s 6.1.1). If a team member did not participate in the inception workshop, however, cross-site visits (s 6.1.2) and mutual rotation (s 6.1.3) during the course of the project provided the team member opportunities to get to know the remote members of the distributed team while working together in close proximity with one another for a short period of time. Cohesion developed within the team when team members know each other (s 4.1.1). Hence, temporary co-location of a distributed team develops cohesion among team members (s 4.1). Unsurprisingly, a distributed team that did not adopt the Same Space strategy was challenged to develop cohesion among the team members:
"We used to work together daily, but we did not know each other. [...] It’s hard to get the feeling of a team when you don’t know who you are working with.” —P9, Developer.

While working together in close proximity with one another during the temporary co-location, the team members understood the importance of engaging in frequent communication and close collaboration (s 4.1.2). During the temporary co-location, face-to-face interaction helped to build rapport among team members. When the team members return to their base site after the temporary co-location, the rapport among them continued to sustain team cohesion:

"...when they returned to the onsite location, they continued to keep a close connection with us at the offshore site.” —P29, Scrum Master.

When members of a distributed team worked together temporarily as a co-located team, they developed a solid shared understanding of the processes and practices that should be in place across the different sites of the distributed team (s 6.1.5). Co-location provided the team members with a clear understanding of what standards, artifacts, technologies, and tools must be used at the different sites of the distributed team for a particular software development project (s 7.2.3).

Same Space strategy primarily affected the Same Team and Same Practices strategies. Hence, we propose a distributed team adopts the Same Space strategy to promote temporary physical co-location among members of the distributed team in order to develop cohesion among the team members and to promote the use of common practices across different sites of the distributed team.
10.2.4 Same Culture strategy

Same Culture strategy aims to promote cross-cultural communication among team members in order to bridge cultural differences from the national and organisational cultural perspectives (s 7.1). Culture, in general, includes notions of shared values, beliefs, expectations, customs, jargon, and rituals (s 2.3.4). Hence, culture affects the way individuals think, act, and communicate with others. In particular, culture in software companies affects the way individuals in a project team use software processes and practices, communicate with others in the team and stakeholders, and react to situations which arise during software development.

A culture of open communication within an organisation developed cohesion among team members (s 7.1.3). Open communication fostered honest, transparent, and direct communication among members of a distributed team, and encouraged close collaboration between the development team and stakeholders (s 4.1.2):

"...this sort of openness is practised throughout our organisation. So, everyone understands how to communicate and collaborate while working in a team." —P11, Scrum Master.

When managers in an organisation treated team members across different sites of a distributed team as equal as possible (s 7.1.4), the sense of oneness among all team members was notably present. Similarly, managers also observed how the whole team worked together rather than how an individual team member performed his work in a team-oriented software development project (s 4.1.4). Hence, cohesion naturally developed among the team members:

"When managers treat them at the same level, all of them feel the togetherness with the team and the organisation." —P7, Agile Coach.
A culture of sharing knowledge with one another in an organisation rooted the practice of writing technical blogs, storing knowledge artifacts in knowledge repositories, and conducting technical presentation for the purpose of sharing knowledge, experience, and skills with the team members and the wider community in the organisation (s 7.2.2):

“In this company, every team, Agile or non-Agile, practices knowledge sharing in one way or another.” —P48, Senior Manager.

With an organisation culture which gave equal importance to all individuals in the organisation, team members came forward to share inconveniences and difficulties that arise from distribution (s 7.2.4). Distributed teams often rotated the team members setting up the communication channel for formal meetings and updating the Wiki with project-context knowledge, and each site of the distributed team took turn to be scheduled for meetings at a time that is less convenient to them.

Same Culture strategy primarily affected the Same Team and Same Practices strategies. Hence, we propose a distributed team adopts the Same Culture strategy to promote cross-cultural communication among members of the distributed team in order to develop cohesion among the team members and to promote the use of common practices across different sites of the distributed team.

10.2.5 Same Practices strategy

Same Practices strategy aims to promote the use of common practices across different sites of a distributed team (s 7.2). Working together using common practices primarily developed cohesion among the team members (s 4.1).
A platform which shares applications used for collaboration is essential to a distributed team (s 7.2.3). While working together using collaborative software, team members coordinated the development tasks through frequent communication with the whole team. Despite distribution, the use of collaborative software encouraged team members to engage with one another (s 4.1.2). Supporting distributed teams with proper tools for collaboration and communication is a strategic initiative for developing cohesion among team members:

“...in our company, we use a number of tools for collaboration, [in which] team members know who’s available for quick a chat or to do remote pair-programming. [...] these tools get them to work together and stay together.” —P20, Senior Manager.

Shared standards such as coding convention and code coverage enabled team members across different sites to be consistent with development processes or practices (s 7.2.1). Using shared standards, the development teams were able to engage with one another, on a common ground, without hesitation (s 4.1.2).

“...as developers, we interact to discuss whether our codes meet the standards or the code coverage is above certain threshold or features meet the requirements.” —P14, Developer.

Same Practices strategy primarily affected the Same Team strategy. Hence, we propose a distributed team adopts the Same Practices strategy to promote the use of common practices across different sites of the distributed team in order to develop cohesion among the team members.
10.3 Applicability of the theory to co-located teams

In this section, we discuss whether the theory of one team which explicates how distributed teams in the global scenario [242] works together, may apply to co-located teams in which the team members work together in close proximity with one another.

Unlike a co-located team, the members of a distributed team face many inconveniences such as working together from different time zones. While some distributions still provides adequate time overlap for synchronous communication, others do not provide any time overlap during the normal working hours of the team members across the different locations.

While working from different time zones, the members of a distributed teams often stretch the normal working hours in order to create a time overlap for synchronous communication between them. For example, one side of the distributed team often starts work before the normal working hours, while the other side stays back after the normal working hours in order to attend joint meetings such as the daily standup meetings or the sprint retrospective meetings. Hence, the Shifting Working Hours (s 5.1.4) is a common tactic for distributed teams but does not apply to co-located teams. Figure 10.2 shows how distributed teams create time overlap.
While members of a distributed team interact with one another using technology-mediated communication tools, often for a limited time due to time zone differences between sites, the members of a co-located team interact face-to-face with one another throughout the normal working hours. For example, the practice of pair-programming can be carried out for both the distributed teams and co-located teams. While pair-programming for a distributed team is dependent on time overlap and technology constraints, this practice can be carried out anytime during the normal working hours for a co-located team. Hence, the tactics in the Same Time strategy (s 5.1) such as the Planned Formal Meeting (s 5.1.1), the Spontaneous Conversation (s 5.1.3), and the Real-time Collaboration (s 5.1.5) that promote synchronous communication between team members happen naturally, without much effort for a co-located team. The Planned Unconference (s 5.1.2) tactic, however, does not apply to co-located teams because members of a co-located team have every opportunity to engage in casual conversation throughout the normal working hours.
The *Same Space* strategy (§6.1) aims to promote temporary physical co-location for the members of a distributed team. Hence, the tactics such as *Rotation* (§6.1.3), *Coach Travels* (§6.1.4), or *Temporary Co-location* (§6.1.5) in this strategy does not apply to co-located teams because the members of a co-located team indeed work together from the same physical space. Meanwhile, the members of a co-located team know one another in person through frequent face-to-face interaction. Hence, the approach of *Knowing the Team* (§4.1.1) which is crucial for distributed teams to “put a name to the face” does not apply to co-located teams.

Unlike distribution, team co-location promotes frequent interaction and close collaboration through face-to-face conversation between team members. Hence, a co-located team may easily develop a shared understanding of the processes and practices that must be in place within the team. Due to this shared understanding, the members of a co-located team use a common set of work practices while carrying out software development activities. The tactics in the *Same Practices* strategy (§7.2) such as the *Shared Standards* (§7.2.1) and *Shared e-Workspace* (§7.2.3) that promote the use of common practices across different sites of a distributed team happen naturally for a co-located team, owing to the shared understanding between the members of the team. A diverse, co-located team may comprise individuals from different cultures, much similar to a distributed team. The tactics in the *Same Culture* strategy (§7.1) such as *Cultural Awareness* (§7.1.1) and *Open Communication* (§7.1.3) may still be useful and applicable for the co-located teams.

### 10.4 Observational experience

In this study, we captured participants’ experience and views on Agile software development through interviews. Most of the interviews were conducted at convenient places, often away from the actual workplace of the participant such as in a meeting room at the company.
When we conducted the interviews with participants, we also wanted to observe the teams in action at the workplaces. We realise observations provide insight into the research phenomena through viewing the project artifacts and experiencing the ongoing team dynamics. During most interviews, however, we were not given permissions to view the workplace or even to have a “peek” at the workplace. The main reason for this was teams were extremely busy with upcoming deadlines and therefore a “tour” to the workplace or observation of the team members in action at the workplace may distract them and subsequently delay their work. Another common reason was an observation at workplace requires special permission from senior management in the organisations and often such permissions were not granted to an “outsider”. Several participants, however, allowed us to conduct proper observations at their workplaces.

In this study, we conducted observations on seven different teams – five in India, one in the USA, and one in Australia. In particular, we observed Scrum practices such as joint standup meetings across different sites, XP practices such as distributed pair programming, the Scrum artifacts (physical or online) such as burndown charts, infrastructures such as video-conferencing rooms, and collaboration tools such as Wikis. We used the observations to check whether what participants say they do is the same as what they actually do. Hence, we were able to validate the authenticity of the interview data.

Several teams kept both the physical and online artifacts for the projects. Figure 10.3 shows the Scrum artifacts for a project. While the physical artifacts provided team members at a particular site with a quick-to-view, easy-to-access experience, the online artifacts provided them with a flexible, integrated, consistent-across-global-sites experience.
One of the teams kept a complete list of stories on a physical task board based on the online task board. Figure 10.4 shows a physical Scrum task board. A task board is a tabular representation of user stories with corresponding tasks of the current sprint. Scrum defines four columns in a task board: (1) “Stories”, which lists the user stories or requirements for a sprint; (2) “To Do”, which lists all tasks belonging to a particular user story that have not been worked on yet; (3) “In Progress”, which lists all tasks belonging to a particular user story that are currently ongoing; and (4) “Done”, which lists all tasks belonging to a particular user story that are completed. Some teams generate all or most of their tasks at sprint planning while others generate tasks as they start a story or just-in-time. A team member assigns a task to himself from the “To Do” column and moves the card into the “In Progress” column. When the task is done, he then moves it into and gave due recognition to the members of a distributed team. the “Done” column and likely choose another task to start.
While many participants discussed the extensive use of communication and collaboration tools, five participants actually showed us the tools used by the teams. We were also shown the online burndown charts. Figure 10.5 shows an online burndown chart using ScrumWorks Pro, which is a Scrum tool developed by CollabNet. A burndown chart is the main measurement metric that is used within Scrum. The burndown chart can be used to determine the velocity of a team. For example, we can know how many story points a team can fulfill within one sprint. Therefore, the team can make a much better estimation for the next sprint.

We were shown in detail how the team used an online project management tool, the Atlassian Jira, in order to capture and organise issues (stories, bugs, features), prioritise the important activities, and stay up-to-date with the activities in the project. Figure 10.6 shows a screen shot of the Atlassian Jira.
Thoughtworks Mingle is another project management tool that offers visibility into project status and collaboration among members of a distributed team. Figure 10.7 shows a screen shot of the Thoughtworks Mingle.
10.5 Selecting a theoretical coding family

A theoretical coding family “conceptualises how the substantive codes will relate to each other as a set of interrelated hypotheses to be integrated into the theory” [103]. Hence, the identification of the most suitable theoretical coding family is essential to integrate the substantive categories into a theory [103, 109, 111].
The use of a theoretical coding family promotes the explanatory power and increases the completeness and relevance of a grounded theory, resulting in a theory with greater scope and parsimony [111]. Hence, Glaser [111] argues a Grounded Theory is best when a theoretical coding family has been used to explicate the research phenomenon. Researcher, however, should be familiar with the theoretical coding families (s 3.7.6) in order to be able to select the most suitable theoretical coding family for the development of a theory [103, 109, 111].

In this study, we explored a number of “candidate” theoretical coding families as possible ways of developing the theory. Next, we describe those “candidate” theoretical coding families and how we finally selected the Strategy theoretical family to explicate our theory.

10.5.1 System Parts

We initially explored the System Parts theoretical coding family [103] in order to understand the relationship between the emergent substantive codes. Figure 10.8 illustrates the System Parts theoretical coding family. System Parts comprises two kinds of theoretical codes: whole and part. This theoretical coding family describes the relationships between parts or subsets of parts to the behaviour of the whole. In particular, System Parts describes how a whole is dependent on its parts that define its behaviour [111]. Glaser explains “parts are necessary but not sufficient, and it is important to organise a substantive theory that shows how a whole is dependent on its part(s) that define its behaviour but neither can exist without the other” [111]. In other words, the behaviour of the whole can only be adequately described through the description of all the parts that form the whole and the interdependent relationship between them. The interdependent relationships between the whole and its parts, however, can become quite complex when parts themselves are interdependent on one another, or when parts effects parts which can effect the whole.
10.5. SELECTING A THEORETICAL CODING FAMILY

We carried out theoretical coding to find out whether the whole can be depicted by the process of *becoming one team*, and the parts can be depicted by the substantive codes: *Same Team, Same Time, Same Space, Same Culture* and *Same Practices*. Meanwhile, “Agile software development” defined the environment and “spatial, temporal, socio-cultural distances” defined the boundaries. We, however, could not establish any interdependent relationships between the parts to the behaviour of the whole. We found each substantive code did not adequately describe the process of becoming one team in Agile software development with distributed teams. Subsequently, we concluded System Part is not the best ‘fit’ for these emergent substantives codes.

### 10.5.2 Six C

We then explored the Six C theoretical coding family in order to understand the relationship between the emergent substantive codes. Figure 10.9 illustrates the Six C theoretical coding family.
The Six C theoretical coding family explicates a core category in terms of six theoretical codes: (1) Context, (2) Condition, (3) Causes, (4) Consequences, (5) Contingencies and (6) Covariances. In the Six C, the context is the ambiance in which the phenomenon occurred; the cause is the reason, source or explanation for the occurrence of the phenomenon; the condition or qualifier is an intervening variable; the covariance is a correlation where one category changes with another; the consequence is the anticipated or unanticipated outcome of the phenomenon and is dependent on cause; and the contingency is the strategy for resolving the problems in the phenomenon. Glaser [103] characterises the Six C as the “bread and butter” theoretical coding family of sociology. Hence, researchers should first keep this theoretical coding family in mind when coding data.

![Theoretical coding family: Six C](image)

Figure 10.9: Theoretical coding family: Six C
10.5. SELECTING A THEORETICAL CODING FAMILY

During theoretical coding, we were able to define the context, condition, causes, and contingencies, but not the covariance and consequences. For example, we attempted to define the context as “Agile software development”, the condition as “Agile teams dispersed across different continents”, the causes as “spatial distance”, “temporal distance”, and “socio-cultural distance”, and contingencies as “Same Team”, “Same Time”, “Same Space”, “Same Culture”, and “Same Practices”. The covariance and the consequences, however, never emerged during the analysis. We attempted to re-analyse the substantive codes, but we still did not “see” the covariance and the consequences emerging during theoretical coding. Hence, we concluded the Six C is not the best ‘fit’ for these substantives codes.

We, however, realised the contingencies describe the strategies which distributed teams continually adopt for bridging the spatial, temporal, and socio-cultural distances in order to become one team. We also realised the causes describe the factors which may influence the adoption of these strategies. Hence, we began to explore the Strategy theoretical coding family in order to understand the relationship between the substantive codes.

10.5.3 Strategy

Continual theoretical coding gave rise to the “strategies” theoretical code for the Same Team, Same Time, Same Space, Same Practices, and Same Culture substantive codes, and the “impact factors” theoretical code for the Trust and Senior Management Support substantive codes (s 3.7.6). When we compared these theoretical codes with the theoretical coding families, we found the Strategy coding family emerged to be the best ‘fit’ for the basic social process of becoming one team and its properties. The Strategy coding family is applied when there exist specific strategies that can be used to resolve a problem in the research phenomenon. Figure 10.10 illustrates the Strategy coding family for the emergent substantives codes.
A complete diagram of "The Theory of One Team" using the Strategy coding family is shown in Figure 4.1. We provided a detailed description of the strategies and impact factors in chapter 4.

### 10.6 Revisiting the Agile Manifesto

In this section, we revisit the Agile Manifesto [24] in order to tie the findings from this study to the core values and principles behind the Agile Manifesto. While the highest priority of an Agile team is to satisfy the customer, it is not surprising to know the members of a distributed, Agile team strived to work together as one team in order to provide business value to the customer.
Despite the distribution, the members of development teams and customers continually engaged with one another (s 4.1.2) in order to develop cohesion within the teams. The team members also established a shared vision (s 4.1.3) to understand the business value that a project brings to customer and worked together effectively towards achieving the project goals. Hence, a cohesive, distributed team with a shared vision worked together effectively and embraced changes to the project requirements in order to provide business value to customer (s 2.1.2: principle #1, #2).

In this study, the development teams for the majority of the projects worked on 2 to 3 weeks sprints and short releases (s 3.6.4) in order to deliver incremental working features of the final software product to customers on a regular basis. Owing to the planned formal meetings (s 5.1.1), the development teams were able to get frequent feedback from customers on requirements and acceptance. Thus, the customer continually gained business value even before the final software product is completed (s 2.1.2: principle #3).

Understanding the importance of close collaboration with customer and working together on a daily basis throughout the project, the members of a development team often stretched the normal working hours (s 5.1.4) in order to create a time overlap which provided them with the opportunities for spontaneous conversation (s 5.1.3) and real-time collaboration (s 5.1.5) with one another (s 2.1.2: principle #4).

Recognising how the whole team performed in a particular sprint or project (s 4.1.4) rather than how an individual performed his job in a team, encouraged the whole team to work together effectively on team-oriented software development activities. Senior management support (s 8.2) in terms of technical infrastructure and customer liaison provides the team with a supportive environment to get the job done (s 2.1.2: principle #5).
Realising the importance of face-to-face conversation between team members, a large number of distributed teams practised cross-site visits (s 6.1.2) and mutual rotation (s 6.1.3) particularly for sharing knowledge and building trust with one another. Whenever possible, the coach preferred to travel (s 6.1.4) to the different sites of a distributed team in order to provide face-to-face coaching to team members. Unsurprisingly, a significant number of distributed teams practised temporary co-location (s 6.1.5) so that team members from different sites can to get to know one another through face-to-face interactions (s 2.1.2: principle #6).

Central to customer satisfaction is the development of a working software. Hence, the members of a distributed team adopted shared standards (s 7.2.1) and worked together using shared e-workspace (s 7.2.3) while managing a shared knowledge (s 7.2.2) across the whole team in order to be able to deliver a working software to customers (s 2.1.2: principle #7).

Emphasizing on developing co-equal team members across different sites of a distribution, the team shared the pain (s 7.2.4) and adopted shared value (s 7.1.4) in order to maintaining a constant progress on a project while keeping the team together (s 2.1.2: principle #8).

With the focus on technical excellence, the team engaged in continuous learning and managed project-context knowledge (s 7.2.2) in order to develop a working software. Meanwhile, the teams also kept an open communication (s 7.1.3) with one another so that the whole team received continual, substantial feedback for improving the processes and practices in place within the team (s 2.1.2: principle #9 and #10).
10.7. EVALUATING OUR THEORY

Self-organising is an important characteristic of an Agile team whereby a team is empowered to take decisions and given the freedom to carry out the software development activities without frequent intervention from managers. Senior management support (s 8.2) is essential to develop a culture for agility to flourish in the team and the wider organisation, which in turn fosters the self-organising nature of the team (s 2.1.2: principle #11).

Despite the distribution, a large number of distributed teams organised planned formal meetings (s 5.1.1) such as the retrospective meetings in order to reflect on how to improve the processes and practices in place within the team and the behaviour of team members on an ongoing basis. Open communication (s 7.1.3) helped the team members to provide direct feedback with one another in order to recognise the strengths and the weaknesses of the whole team (s 2.1.2: principle #12).

In summary, distributed teams in Agile software development adopted strategies which aligned with the core values and principles behind the Agile Manifesto to bridge the spatial, temporal, and socio-cultural distances in order to become one team.

10.7 Evaluating our theory

Glaser and Strauss [114] originally proposed Grounded Theory as an alternative method to the approaches of theorising that dominate sociological research, namely quantitative testing and verification of a “dreamed-up, speculative, or logically deduced theory”. Glaser and Strauss [114] advocate the Grounded Theory method is for building theory, not verifying it. Hence, follow-up studies for verification of emergent theories are left to other research methods that are better suited to that purpose [108]. Glaser [108] argues a theory which is grounded in data, in some sense has already been verified:
“A grounded theory gets its concepts from the data; it does not bring ideas to force the data, that need to be subsequently tested”.

In this section, we primarily discuss how Glaser’s [103, 108] criteria for evaluating the credibility of a grounded theory and Angen’s [14] validation approaches for interpretive, qualitative research were used to evaluate our grounded theory. We also discuss how well our grounded theory evaluates against Lincoln and Guba’s [189] criteria of trustworthiness for evaluating a qualitative research study and Weber’s [308] criteria for evaluating the quality of a theory.

10.7.1 Glaser’s approach

Glaser [103, 108] suggests evaluating the credibility of a grounded theory through four criteria: *fit, work, relevance,* and *modifiability* (s 3.9). A core category that fits, works, is relevant, and is subject to continual modification, will integrate a theory in the way that the theory is dense and saturated with relationships.

In this study, we avoided reading extensive literature from the same substantive area before commencing the research study (s 3.5.4). Our knowledge on the key concern of distributed teams in Agile software development is primarily acquired through interaction with participants while collecting data from them (s 3.6.3). We particularly employed theoretical sampling, in which the decisions on what data to collect and where to find them were determined based on the emerging theory (s 3.6.1). We managed to select suitable participants from different roles across different software companies in different countries who provided us with relevant explanations on the emerging theory from different perspectives (s 3.6.4). We also continually changed the interview questions in order to focus on the concerns that seem central to the emerging theory.
10.7. EVALUATING OUR THEORY

Theoretical sampling was continued until complete redundancy, or theoretical saturation, was noticed in the new data. Therefore, the categories and their properties which define the theory have only emerged from the data and were not forced by preconceived ideas or hypothesis. Hence, we argue our theory fits the realities under study.

One way to find out the “work” and “relevance” of the theory is to ask the participants whether the core category relates to the salient social problem in the area studied and the theory explicates the pattern of behaviour used to address the problem. When and as categories emerged from data analysis, we attempted to write technical papers explaining the emerging theory. We then provided the participants with the technical papers and requested them to give feedback on the emerging theory. Here’s an excerpt of the written feedback received from a participant:

“...your research explains, in a wider perspective, what practices teams should adopt and keep doing, and what practices they should avoid doing.” – P4, Agile Coach.

Subsequently, we published these technical papers and presented them at international conferences (see Appendix A). To date, we have presented different aspects of the emerging theory to the participants and the wider Agile community at various international conferences in the USA, Norway, Spain, Sweden, Italy, India, Australia, and New Zealand. We received continual feedback from practitioners describing how the emerging theory has been useful, meaningful, and applicable to them. Excerpts of the written feedback received from practitioners after the presentations at international conferences follow.

“[Company] has worked as a distributed organization for several years, but I can see there is a lot we can learn from your research.” – Practitioner, USA (via email).
“Your talk was so important, especially as the reality of distributed Agile teams starts to sink in with a number of larger, high budget projects. Your findings very much support my experience on the distributed team I’m on right now.” – Practitioner, USA (via email).

Hence, we argue our theory works and is relevant for practitioners working in the substantive area.

Our theory has evolved over time when we compared new data to the existing concepts and categories. Therefore, we believe our theory can still continue to evolve when new relevant data is compared to the existing theory. Hence, we argue our theory is readily modifiable when new data present variations in the emergent substantive codes.

10.7.2 Angen’s approach

In this study, we have also adopted Angen’s [14] validation approach for interpretive, qualitative research: ethical validation and substantive validation (s 3.9). We focused on validation when research is formulated, carried out and written up. Thus, we adopted a systematic set of precise procedures for collection, analysis, and articulation of conceptually abstract theory.

Owing to the traditional approach to research, some Grounded Theory researchers still conduct extensive literature review before commencing a research study:

“I use literature extensively when I conduct research, as I collect, code, memo and write. This broad reading of literature is another of my deviations from the proscriptions of classical Grounded Theory” [187].
In contrast, we avoided a major upfront literature review in the substantive area when the research is formulated in order to prevent generating a focus from the literature rather than from the data (s 3.5.4). We, however, integrated literature into the emerging theory through the data analysis process after the core category, its properties, and all other related categories have emerged. We employed theoretical sampling until we reached theoretical saturation (s 3.6.1), and continually wrote theoretical memos on the code, concept and categories in order to develop a conceptually dense substantive theory (s 3.7.4). We finally performed theoretical sorting in order to write up the theory (s 3.7.5). Hence, in order to achieve ethical validation, we used classic Grounded Theory processes from inception to completion of research.

In this thesis, we self-disclosed personal assumptions, beliefs, and biases that shaped the research, right from the beginning of the research (s 3.4). We also included selected interview quotations throughout the results chapters in order to describe how the categories and their properties emerged from data analysis (s 3.7.1). Most importantly, we included a complete coding structure showing the codes, concepts and categories that emerged from data analysis in order to document evidence on how the theory was developed in this research (see Appendix D). Hence, in order to achieve substantive validation, we provided adequate description pertaining the research for others to judge the trustworthiness of the meanings arrived at in the end.

10.7.3 Lincoln and Guba’s approach

In this section, we discuss how well our theory evaluates against Lincoln and Guba’s [189] criteria of trustworthiness. Credibility, which is the most important criteria in establishing trustworthiness, was achieved through triangulation of data, peer debriefing, and member checking:
CHAPTER 10. DISCUSSION

Triangulation of data: We primarily conducted interviews to gather data from participants (s 3.6.3). Using theoretical sampling, we interviewed participants from different roles such as developers, business analysts, product owners, and senior managers in order to acquire a comprehensive view of the research phenomenon (s 3.6.1). With participants coming from different software companies across different continents, we got to know the social problem in the area studied and the pattern of behaviour used to address the problem from different perspectives. We also conducted several observations to validate the authenticity of the interview data (s 3.6.3). Since literature is perceived as another source of data, we decided to review the literature when the emerging concepts and categories were sufficiently grounded, and then integrate them into the emerging theory (s 3.5.4). To summarise, we used a wide range of participants from different sites, and different data collection techniques in order to triangulate the data used in this study.

Peer debriefing: We further added credibility to the study with the assistance of peer debriefers. The primary researcher continually discussed the emerging theory and the Grounded Theory processes with his PhD supervisors and mentors such as Dr. George Allan primarily for getting feedback to enhance credibility and ensure validity. These peer debriefers helped to clarify interpretations and challenged researcher assumptions, contributing to the credibility of the emerging theory.

Member checking: Though the interviews with participants were voice recorded, we subsequently transcribed them for use during data analysis. We followed a number of steps to verify the data collected from the participants. We provided participants with the respective interview transcriptions and observational field notes, and requested them to verify the data. We encouraged participants to correct errors and challenge what are perceived as wrong interpretations, and to provide further information
deemed important for them. We also provided the participants with the re-
search papers which we published at conferences in order to acquire feed-
back on the emerging theory. We treated the feedback as another source
of data that can be integrated into the emerging theory through the data
analysis process.

The dependability/auditability criteria is much similar to Angen’s ethical
validation (s 10.7.2). In this thesis, we reported in detail how initial data
was collected through selective sampling and how the subsequent data
was collected through theoretical sampling (s 3.6.1). We then described in
detail how data analysis was carried out from the beginning to the end
(s 3.7). We also provide a complete coding structure (with codes, concepts,
and categories) for our theory (see Appendix D).

We achieved confirmability by doing a “pure” Grounded Theory study,
in which we focus on the experiences and ideas of participants, rather than
the preferences of the researcher. We carried out a minor literature re-
view at the beginning of the study in order to prevent generating a focus
from the literature rather than from the data (s 3.5.4). We thus can be sure
that the concerns emerging from data are an actual reflection of that data
rather than preconception based on the literature. The major literature
review was conducted when the theory seems sufficiently grounded and
developed in order to relate the theory to the existing literature. During
data collection, we emphasised the anonymous status of the participants
and invited them to offer data, freely, willingly, and without fear of losing
credibility in the eyes of managers of the organisation. We have also made
it clear to participants that they have the right to withdraw from the study
at any point, without a need to disclose an explanation to us. By doing so,
we hoped to ensure honesty in participants when contributing data.
In this thesis, we provided a detailed description of the participants, projects, and teams in order to achieve transferability (s 3.6.4). With adequate information on research context, another researcher could employ the Grounded Theory’s constant comparison method on data from new contexts to determine whether substantive findings fits the new data, or the findings are transferable to another context.

10.7.4 Weber’s approach

Weber [308] suggests evaluation should focus on the quality of the individual parts that make up the theory and the quality of the theory as a whole: constructs, associations, states, and events constitutes the parts, whereas importance, novelty, parsimony, level, and falsifiability constitutes the whole (s 3.9). Figure 10.11 shows how the parts from Weber’s framework is perceived in Grounded Theory.

Parts

In this study, categories and their properties represents classes and their constructs (s 3.9). Therefore, the exact nature of the things that a theory covers will be clear when the categories and their properties have been defined precisely. Hence, we have given precise definition of the substantive and theoretical codes in chapters 4–8. Figure 10.12 shows the list of definition of the substantive and theoretical codes.

We have also given precise definition of the key terms used to explain our theory. Figure 10.13 shows the list of definition of the terms used to explain our theory. We particularly provided precise definition of all the constructs that our theory covers so that the association between them will be clear and meaningful.
In this study, conceptual relationships between the emergent categories represents associations (s 3.9). During theoretical coding, constant comparative analysis gave rise to two theoretical codes – “strategies” and “impact factors” – which describe the conceptual relationships between substantives codes (s 3.7.6). The theoretical code “strategies” describes the conceptual relationship between the substantive codes Same Team, Same Time, Same Space, Same Culture, and Same Practices, whereas the theoretical code “impact factors” describes the conceptual relationship between the substantive codes Trust and Senior Management Support. Continual analysis revealed a conceptual relationship between these theoretical codes. We particularly found the “impact factors” influence the “strategies”.
## CHAPTER 10. DISCUSSION

<table>
<thead>
<tr>
<th>Class Type</th>
<th>Class Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substantive</td>
<td>Same Time</td>
<td>The Same Time strategy aims to promote synchronous communication in order to bridge temporal distance.</td>
</tr>
<tr>
<td></td>
<td>Same Space</td>
<td>The Same Space strategy aims to promote temporary physical co-location in order to bridge spatial distance.</td>
</tr>
<tr>
<td></td>
<td>Same Team</td>
<td>The Same Team strategy aims to develop team cohesion between members of a distributed team.</td>
</tr>
<tr>
<td></td>
<td>Same Culture</td>
<td>The Same Culture strategy aims to promote cross-cultural communication in order to bridge cultural differences.</td>
</tr>
<tr>
<td></td>
<td>Same Practices</td>
<td>The Same Practices strategy aims to promote the use of common practices across different sites of a distributed team.</td>
</tr>
<tr>
<td></td>
<td>Trust</td>
<td>A willingness to rely on another party and to take action in circumstances where such action makes one vulnerable to the other party.</td>
</tr>
<tr>
<td></td>
<td>Senior Management Support</td>
<td>A form of organisational, technological, or financial assistance from senior management in an organisation.</td>
</tr>
<tr>
<td>Theoretical</td>
<td>Strategy</td>
<td>A high-level plan of action designed to achieve a particular goal.</td>
</tr>
<tr>
<td></td>
<td>Impact Factor</td>
<td>A construct that influence the adoption of a strategy.</td>
</tr>
</tbody>
</table>

Figure 10.12: Definition of substantive and theoretical codes.
### 10.7. Evaluating Our Theory

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile (s 2.1)</td>
<td>An umbrella term for a software development philosophy that encompasses the values, principles, methods, and teams.</td>
</tr>
<tr>
<td>Agility (s 2.1.4)</td>
<td>The ability of a team to embrace changes at any stage of a project in order to deliver working software that provides business value to customer.</td>
</tr>
<tr>
<td>Agile methods (s 2.1.3)</td>
<td>A family of software development methods that follow an iterative and incremental style of software development.</td>
</tr>
<tr>
<td>Project team (s 2.1.4)</td>
<td>A group of committed, interdependent individuals with complementary skills who work together in order to complete a software development project.</td>
</tr>
<tr>
<td>Agile team (s 2.1.4)</td>
<td>A cross-functional, self-organising, high-performance team that carries out software development activities in line with the principles behind the Agile Manifesto in order to provide business value to customer.</td>
</tr>
<tr>
<td>Distributed team (s 2.2.1)</td>
<td>A project team from the same organisation where at least one member of the team is physically distant from the others in different countries, across different continents.</td>
</tr>
<tr>
<td>Temporal distance (s 2.2.2)</td>
<td>A measure of the temporal dispersion, occurring when team members wishing to interact.</td>
</tr>
<tr>
<td>Spatial distance (s 2.2.2)</td>
<td>A measure of the geographical dispersion, occurring when team members are dispersed across different sites.</td>
</tr>
<tr>
<td>Socio-cultural distance (s 2.2.2)</td>
<td>A measure of the effort required by team members to understand the national and organisational cultures across sites.</td>
</tr>
<tr>
<td>The Theory of One Team (s 4)</td>
<td>An emergent grounded theory which explains how a distributed team in Agile software development adopts explicit strategies for bridging spatial, temporal, and socio-cultural distances, while facing critical impact factors, in order to become one team.</td>
</tr>
</tbody>
</table>

Figure 10.13: Definition of terms.
We found our theory covers a dynamic phenomena because the core category is a basic social process of becoming one team. Weber [308] suggests “an arrow is placed on the association” when using a diagram to represent the association between constructs in a theory that covers dynamic phenomena. Hence, we used arrows in figure 3.13, figure 3.14 and figure 4.3 to show the conceptual relationships that exist in our theory.

We specified clearly and as precise as possible the things within the research phenomenon, or the state space, for which our theory is intended to cover (s 3.9). For example, we have described the area of research (s 3.5.3), the type of team investigated (s 2.2.1), and the global distances (s 2.2.2). Most importantly, we have provided a detailed description of participants, teams, and projects investigated in this study (s 3.6.4).

In dynamic phenomena, events refer to happenings that change over time such as the basic social process of becoming one team (s 3.9). A basic social process explains the preponderance of behaviour in a substantive area, occurs over time, and involves change over time in the light of new data (s 3.7.2). A theory that covers basic social process should clearly articulate the pattern of behaviour about what is going on in the area studied. In this thesis, we presented the theory of one team which, in particular, describes how members of a distributed team adopt a strategy or a combination of strategies to bridge spatial, temporal, and socio-cultural distances in order to become one team.

Next, we continue to evaluate the attributes of the theory as a whole. In particular, we evaluate the importance, novelty, parsimony, level, and falsifiability of our theory.
Whole

A co-located Agile team focuses on close collaboration, frequent interaction, and face-to-face communication as ways to work together with one another in the team in order to deliver valuable software to customer on a regular basis (s 2.1.2). A distributed team, however, unavoidably experience major problems related to communication, coordination, and collaboration due to geographical, temporal, and socio-cultural dispersion of the team members (s 2.2.2). Many studies [153, 161, 230, 280] suggest there is a critical need for an in-depth study on integrating Agile methods with distributed software development in order to understand the overall combination of the two approaches. However, there is still no deep, theoretically grounded, understanding of the applicability of Agile methods to distributed software development [280].

We carried out a large-scale, empirical research study in order to investigate how the members of a distributed team in Agile software development work together (s 3.6.4). In this study, we found a set of strategies which distributed teams across 38 different software companies adopt for bridging spatial, temporal, and socio-cultural distances in order to become one team. This thesis describes the most important concern of distributed teams in Agile software development and provides guidance to the members of a distributed team on how to bridge spatial, temporal, and socio-cultural distances in order work together as one team. Our theory addresses an interesting phenomenon which is deemed important from the viewpoint of practice and research. Weber [308] suggests the importance of a theory is a key factor for evaluating the quality of the theory. Hence, we argue our theory meets Weber’s [308] importance criterion.

Weber [308] suggests a theory will be deemed novel when the theory’s focal phenomena have not been covered by prior theories. Based on the extensive literature review carried out while writing this thesis, we re-
alised there is no existing theory which describes how a distributed team in Agile software development become one team. Hence, our theory’s focal phenomenon is original and significantly contributes to the larger body of knowledge. We have published different aspects of our theory in major conferences in the area of Agile software development and software engineering (see Appendix A). Upon completing this thesis, we plan to publish our theory in major journals.

A theory is parsimonious when a small number of constructs have been used to explain the focal phenomena (s 3.9). We argue our theory meets Weber’s [308] parsimony criterion because we articulated our theory using only seven substantive codes for the strategies and two theoretical codes for the impact factors, in line with the “magical number seven, plus or minus two” [211] used to describe what constitutes a “small number”.

In this study, we produced a middle-range, substantive theory, which neither covers a very narrow, constrained set of phenomena nor a very broad range of phenomena (s 3.8). A middle-range theory avoids the “narrow-empiricism” and “theory of everything” criticisms. The theory of one team describes how distributed teams in particular works together in Agile software development. We do not claim our theory to be universally applicable but rather it accurately describes the research phenomena, and it can be further extended when data from new contexts is compared to the existing theory. Hence, we argue our theory meets Weber’s [308] level criterion.

Weber [308] suggest to undertake empirical tests in order to falsify theories. Glaser [108] argues a grounded theory does not require further testing because it comes directly from the data itself. The purpose of the classic Grounded Theory (s 3.5) is not theory testing but rather discovering a theory that aids understanding and action in the area under study:
“Keep in mind that the purpose of grounded theory is not to make truth statements about reality, but, rather, to elicit fresh understandings about patterned relationships between social actors and how these relationships and interactions actively construct reality” [114].

Schwandt [267] argues an emergent theory has been tested and re-tested using theoretical sampling:

“The testing of the emergent theory is guided by theoretical sampling” [267].

At the initial stage of the Grounded Theory process, the emphasis is on induction, or emergence, with the researcher moving from the data to empirical generalisation and on to theory [123, 267]. When the core category emerges, grounded theory shifts from its initial emphasis on induction to deduction, as ideas originally developed by induction from the initial data are constantly re-analysed in the light of further empirical data in order to validate or negate them. Thus, the emergent theory has been built from data, and tested against further data using theoretical sampling. In the section 11.3, however, we suggested a follow-up study may be undertaken for verification of our theory using other research methods that are better suited to that purpose.
Chapter 11

Conclusion

In this chapter, we summarise the main contribution of this thesis – a grounded theory of one team. We then critique our grounded theory and discuss the limitations of this study. We finally present suggestions for future work.

11.1 Research Contribution

The grounded theory of one team presented in this thesis is the first large-scale Grounded Theory study which combines Agile software development and distributed software development. We investigated 52 different software development projects, in which we conducted seven field observations and interviewed 55 participants from 38 different software companies in the USA, India, and Australia. In this study, we discovered five explicit strategies which a distributed team in Agile software development adopts for bridging the spatial, temporal, and socio-cultural distances in order to become one team. We also discovered two crucial factors which influence the adoption of the strategies in distributed teams. This thesis proposes “The Theory of One Team” to explicate the findings from this research study. The contributions of this thesis follows.
1. A set of five strategies, namely Same Team, Same Time, Same Space, Same Culture and Same Practices, for bridging spatial, temporal and socio-cultural distances:

- **Same Team** strategy aims to develop cohesion among members of a distributed team through Knowing the Team, Engaging with the Team, Establishing Shared Vision, and Team Recognition. A detailed description of this strategy is provided in chapter 4.

- **Same Time** strategy aims to promote synchronous communication in order to bridge the temporal distance through Planned Formal Meetings, Planned Unconference, Spontaneous Conversation, Shifting Working Hours, and Real-time Collaboration. A detailed description of the Same Time strategy is provided in chapter 5.

- **Same Space** strategy aims to promote temporary physical co-location in order to bridge the spatial distance through Inception Workshop, Cross-site Visits, Rotation, Coach Travels, Temporary Co-location. A detailed description of the Same Space strategy is provided in chapter 6.

- **Same Culture** strategy aims to promote cross-cultural communication through Cultural Awareness, Language Support, Open Communication and Shared Value.
11.1. RESEARCH CONTRIBUTION

- **Same Practices** strategy aims to promote the use of common practices through Shared Standards, Shared Knowledge, Shared e-Workspace, and Shared Pain. The Same Team, Same Culture, and Same Practices strategies aim to bridge the socio-cultural distance. A detailed description of the Same Culture and the Same Practices strategies is provided in chapter 7.

2. A set of two factors, namely Trust and Senior Management Support, which influence the adoption of the strategies:

- **Trust** is an important factor which influences the adoption of the strategies for bridging spatial, temporal and socio-cultural distances. In this thesis, we present seven techniques for building trust in distributed teams. A detailed description of the importance of trust for Agile teams is provided in section 8.1.

- **Senior management support** is another important factor which influences the adoption of the strategies for bridging spatial, temporal and socio-cultural distances. A distributed team needs support from senior management in terms of organisational culture, human resource management, financial sponsorship, technical infrastructure, and customer liaison. A detailed description of the importance of senior management support is provided in section 8.2.

3. **Agility in context** describes how a distributed team adopts a contextual approach to selecting a strategy or a combination of strategies which are contextually relevant to the team. A detailed description of agility in context is provided in chapter 9.
In this thesis, we also present a detailed description of how the Grounded Theory research method has been used in this software engineering research study. In particular, we describe the Grounded Theory research method in terms of theoretical sampling, open coding, core category, selective coding, theoretical memoing, theoretical sorting, theoretical coding, and generating a theory. We present an example of analysis from interview transcript to a category in order to describe the outcomes (such as code, concept, and category) and the artifacts (such as memos) that emerged in this study. This thesis provides further evidence that the Grounded Theory method can be applied successfully to the field of software engineering. Finally, this research has resulted in a number of publications focusing on various aspects of the theory of one team (see Appendix A).

11.2 Limitations

As we employ Grounded Theory in empirical software engineering research, we often encounter criticism regarding the generalisability of the findings. A Grounded Theory research study produces a “mid-range” theory, which means the theory can be modified with further analysis using new data from the same context or new context. We do not claim our findings are universally generalisable to different contexts. Our findings, however, accurately characterise the contexts studied. In this thesis, we present detailed description of the contexts of our research (§ 3.6.4) so that another researcher could employ the Grounded Theory’s constant comparison method on new data to determine whether a substantive findings fits the new data, or the findings are transferable to another context.

While qualitative data is known to be prone to bias, it is important to clarify what bias the researcher brings to the study during the data collection process [226]. Hence, we clarified our personal assumptions and be-
liefs in this thesis (s 3.4). Researchers often develop bias while embracing a hypothesis too strongly [226]. This research was carried out with only a minor literature review (chapter 2) before the start of the study. Rather than reading existing literature to understand the research phenomenon, we instead focused on collecting data from participants and subsequently analysing them to develop a theory without a preconceived notion of the research phenomenon. The major literature review (s 4.3, s 5.2, s 6.2, s 7.3, s 8.3, s 9.6) was carried out only after each emergent category has been developed and sufficiently grounded.

As this research study has focused on participants from the USA, India and Australia, we acknowledge the findings are biased towards the views of these participants, and must be interpreted as such. We have described how participants were recruited for this study (s 3.6.2).

Another common criticism of employing Grounded Theory in an empirical software engineering research is regarding validity of the emergent theory [6]. Glaser [109] states the focus of Grounded Theory is the generation of theory, and validity of the emergent theory may be undertaken by another researcher using different research methods. We, however, evaluated the credibility of a grounded theory using four criteria – fit, work, relevance, and modifiability – as criteria for quality judgement. We also evaluated our grounded theory against Weber’s [308] framework and criteria for evaluating quality of theories using (1) “Parts” – the evaluation focused on the quality of the individual components that make up the theory; and (2) “Whole” – the evaluation focused on the quality of the theory considered in toto. In addition to Glaser’s [103, 108] and Weber’s [308] evaluation criteria, we have adopted Angen’s [14] validation approach for interpretive, qualitative research – ethical validation and substantive validation – for the validation of our grounded theory. Finally, we have also adopted Lincoln and Guba’s [189] criteria of trustworthiness for evaluating qualitative research studies (s 10.7).
11.3 Future Work

The grounded theory of one team generated in this research is deemed novel because the theory’s focal phenomenon has not been covered by prior theories. Hence, this study raises a number of interesting avenues for future research:

- This research study has produced a mid-range substantive theory within the context studied. This study can be replicated within different contexts such as non-agile projects or outsourced, distributed teams. The findings emerging from the replicated study can be compared against our findings. In this way, researchers may generate a formal theory to explicate the key concern of distributed, software development teams in general.

- The theory of one team could be validated using quantitative approaches in order to verify the emergent concepts and categories presented in this thesis. For example, a survey research can be carried out to investigate how many teams adopt each of the tactics in each of the strategies presented in this thesis.

- In this study, we found senior management support is crucial for distributed teams. Further research may be carried out specifically to investigate the covariance between senior management support and the success of a software development project. Research may also be carried out to investigate how a distributed team secures support from senior management when there is no or little support given to the team.
Appendix A: List of Publications.


Appendix B: Approved HEC Application.
HUMAN ETHICS COMMITTEE
Application for Approval of Research Projects
Please write legibly or type if possible. Applications must be signed by supervisor (for student projects) and Head of School

Note: The Human Ethics Committee attempts to have all applications approved within three weeks but a longer period may be necessary if applications require substantial revision.

1 NATURE OF PROPOSED RESEARCH:
   (a) Staff Research / Student Research (delete one)
   (b) If Student Research Degree: PhD (Software Engineering)
       Course Code: SWEN 690
   (c) Project Title: Distributed Agile Projects.

2 INVESTIGATORS:
   (a) Principal Investigator
       Name: Siva Kumar Dorairaj
       E-mail address: Siva.Kumar.Dorairaj @ ecs.vuw.ac.nz
       School/Dept/Group: Engineering & Computer Science/Software Engineering
   (b) Other Researchers Name Position
       None.
   (c) Supervisor (in the case of student research projects)
       Prof. James Noble and Dr. Petra Malik

3 DURATION OF RESEARCH
   (a) Proposed starting date for data collection: 1 August 2009.
       (Note: that NO part of the research requiring ethical approval may commence prior to approval being given)
   (b) Proposed date of completion of project as a whole: 1 March 2012.
4 PROPOSED SOURCE/S OF FUNDING AND OTHER ETHICAL CONSIDERATIONS

(a) Sources of funding for the project

Please indicate any ethical issues or conflicts of interest that may arise because of sources of funding (e.g. restrictions on publication of results)

None.

(b) Is any professional code of ethics to be followed?

☐ Yes  ☑ No

If yes, name:

(c) Is ethical approval required from any other body?

☐ Yes  ☑ No

If yes, name and indicate when/if approval will be given:

5 DETAILS OF PROJECT

Briefly Outline:

(a) The objectives of the project:

- To explore and understand the nature of distributed agile projects more objectively.

- To identify best practices and information dissemination approaches in distributed agile projects, and the variety of skills required in order to efficiently and effectively deliver quality software and customer satisfaction.

- To propose a contemporary framework that helps the software engineering project management teams to improve performance of distributed agile projects, and recognise the economic, technological and organisational conditions required to undertake any globally distributed agile project.

(b) Method of data collection

- Using qualitative research methodology, such as Grounded Theory, the data required for the research will be systematically gathered using semi-structured interviews, surveys, questionnaires, and observations.

- Research requires data gathering from various software engineering project management teams in New Zealand and across the globe to comprehensively acquire information about distributed agile projects.
(c) The benefits and scientific value of the project

- To increase understanding of distributed agile projects scientifically rather than referring to discrete user experience reports.
- To be able to manage future distributed agile projects successfully by establishing a strong rapport amongst onshore and offshore team members, building efficient teamwork, and developing trust amongst stakeholders.
- The research will construct a deep understanding of subtle but crucial issues involved around distributed agile projects such as communication style, conflict handling tactics and team management (especially if team members are scattered across several time zones, cultures and languages).

(d) Characteristics of the participants

- Software engineering project managers, software developers and team leaders, agile practitioners and consultants who are (directly or indirectly) involved with Agile Project Management.

(e) Method of recruitment

- Organisations and agile practitioners will be identified from agile community lists and user groups (i.e. Agile Alliance / Agile Professionals), and conference participants (Agile200x / XP200x). The researcher will initially approach the potential participants through e-mails and teleconferences, and provide them a brief outline of the research.
- Subsequently, the researcher will actively collaborate with the interested individuals and organisations to develop a mutually agreeable terms for their participation in the research. A research document describing the purpose and scope of the research, the collection and use of the data, feedback mechanisms, and other commitments expected from the participants will be developed and distributed to the participants for their consent.
- Participants nominated (by the respective organisations) will be approached for the research study.
- The initial draft of the consent document, information sheet and interview guide is attached to this application.

(f) Payments that are to be made/expenses to be reimbursed to participants

None.

(g) Other assistance (e.g. meals, transport) that is to be given to participants

None.
(h) Any special hazards and/or inconvenience (including deception) that participants will encounter

None.

(i) State whether consent is for (delete where not applicable):

(i) the collection of data
(ii) attribution of opinions or information
(iii) release of data to others
(iv) use for a conference report or a publication
(v) use for some particular purpose (specify)

Attach a copy of any questionnaire or interview schedule to the application:

Attached.

(j) How is informed consent to be obtained (see sections 4.1, 4.5(d) and 4.8(g) of the Human Ethics Policy)

(i) the research is strictly anonymous, an information sheet is supplied and informed consent is implied by voluntary participation in filling out a questionnaire for example (include a copy of the information sheet)

☐ Yes  ✔ No

(ii) the research is not anonymous but is confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet)

✔ Yes  ☐ No

(iii) the research is neither anonymous or confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet)

☐ Yes  ✔ No

(iv) informed consent will be obtained by some other method (please specify and provide details)

☐ Yes  ✔ No

With the exception of anonymous research as in (i), if it is proposed that written consent will not be obtained, please explain why

Not Applicable.

(k) If the research will not be conducted on a strictly anonymous basis state how issues of confidentiality of participants are to be ensured if this is intended. (See section 4.3.1(e) of the Human Ethics Policy). (e.g. who will listen to tapes, see questionnaires or have access to data). Please ensure that you distinguish clearly between anonymity and confidentiality. Indicate which of these are applicable.

(i) access to the research data will be restricted to the investigator

✔ Yes  ☐ No
(ii) access to the research data will be restricted to the investigator and their supervisor (student research)

☑ Yes ☐ No

(iii) all opinions and data will be reported in aggregated form in such a way that individual persons or organisations are not identifiable

☑ Yes ☐ No

(iv) Other (please specify)

Not Applicable.

(l) Procedure for the storage of, access to and disposal of data, both during and at the conclusion of the research. (see section 4.12 of the Human Ethics Policy). Indicate which are applicable:

(i) all written material (questionnaires, interview notes, etc) will be kept in a locked file and access is restricted to the investigator

☑ Yes ☐ No

(ii) all electronic information will be kept in a password-protected file and access will be restricted to the investigator

☑ Yes ☐ No

(iii) all questionnaires, interview notes and similar materials will be destroyed:

(a) at the conclusion of the research

☐ Yes ☑ No

or (b) three (3) years after the conclusion of the research

☑ Yes ☐ No

(iv) any audio or video recordings will be returned to participants and/or electronically wiped

☑ Yes ☐ No

(v) other procedures (please specify):

Access rights to investigator and his supervisors.

If data and material are not to be destroyed please indicate why and the procedures envisaged for ongoing storage and security

Not Applicable.

(m) Feedback procedures (See section 7 of Appendix 1 of the Human Ethics Policy). You should indicate whether feedback will be provided to participants and in what form. If feedback will not be given, indicate the reasons why.

- The participants will be continuously updated about the research results and all essential findings, and provided with prompt feedback to confirm and review the researcher’s interpretation of their comments/data.
(n) Reporting and publication of results. Please indicate which of the following are appropriate. The proposed form of publications should be indicated on the information sheet and/or consent form.

(i) publication in academic or professional journals

[ ] Yes  [ ] No

(ii) dissemination at academic or professional conferences

[ ] Yes  [ ] No

(iii) deposit of the research paper or thesis in the University Library (student research)

[ ] Yes  [ ] No

(iv) others (please specify)

None.
Signature of investigators as listed on page 1 *(including supervisors)* and Head of School.

**NB:** All investigators and the Head of School must sign before an application is submitted for approval.

……………………………………………   Date………………………...
……………………………………………               Date………………………...
……………………………………………   Date………………………...
……………………………………………   Date………………………...
……………………………………………   Date………………………...

Head of School:

……………………………………………   Date ………………………..
MEMORANDUM

TO      Siva Kumar Dorairaj  
COPY TO Professor James Noble and Dr Petra Malik, Supervisors  
FROM    Dr Allison Kirkman, Convener, Human Ethics Committee  
DATE    June 20, 2009  
PAGES   1  
SUBJECT Ethics Approval: No 16613, Distributed Agile Projects

Thank you for your application for ethical approval, which has now been considered by the Standing Committee of the Human Ethics Committee.

Your application has been approved from the above date and this approval continues until 1 March 2012. If your data collection is not completed by this date you should apply to the Human Ethics Committee for an extension to this approval.

Best wishes with the research.

Allison Kirkman
Convener
Appendix C: Research Documents.

Attachment A: Cover Letter for Research Participant.
Attachment B: Information Sheet.
Attachment C: Consent for Research Participation.
Attachment D: Interview Guide.
Attachment E: Interview Questions.
Attachment A: Cover Letter for Research Participant

[Participant’s Name]
[Company Name]
[Company Address]

Dear [Participant’s Name],

**PhD Research into Distributed Agile Projects.**

Thank you for indicating, in our recent e-mail / teleconference, that you are interested in participating in this research.

Prior to conducting any interview with you, Victoria University of Wellington requires that I obtain your written informed consent, which is a normal part of any research study.

Attached herewith for your kind information:
- A copy of the *research document* that outlines the purpose, scope and approach of the research study.
- An *interview agenda* that describes the issues to be discussed with you.
- A *consent form* that you need to sign and return to me [in the enclosed envelope] if you decide to participate in this research.

I would like to emphasize that you do not have to participate in this research and that you are free to withdraw from this research without any explanation up to one month after the interview.

If you require any further information or if you prefer not to be interviewed, please do not hesitate to contact me. Alternatively, you may also contact my PhD supervisors Prof. James Noble ([kjx@ecs.vuw.ac.nz](mailto:kjx@ecs.vuw.ac.nz) or +64 4 463 6736) or Dr. Petra Malik ([Petra.Malik@ecs.vuw.ac.nz](mailto:Petra.Malik@ecs.vuw.ac.nz) or +64 4 463 5820).

Thanking you in advance.

Yours truly,

Siva Kumar Dorairaj
Mobile: +64 21 1733044
E-mail: Siva.Kumar.Dorairaj@ecs.vuw.ac.nz
Attachment B: Information Sheet

General Information

This research study contributes towards the requirements for a PhD degree at Victoria University of Wellington, New Zealand.

Research Topic: Distributed Agile Projects

PhD Student: Siva Kumar Dorairaj (Siva.Kumar.Dorairaj@ecs.vuw.ac.nz, +64 21 1733044)

Supervisors: Prof. James Noble (kjx@ecs.vuw.ac.nz, +64 4 4636736)
Dr. Petra Malik (Petra.malik@ecs.vuw.ac.nz, +64 4 463 5820)

Objectives of Research

The objective of this research is to explore and understand the nature of distributed agile projects more objectively in order to identify the best practices and the variety of skills required to efficiently and effectively deliver quality software and customer satisfaction. This research also aims to propose a contemporary framework that will help the software engineering project management teams improve the performance of distributed agile projects, and recognise the economic, technological and organisational conditions required to undertake any distributed agile project.

Research Methodology

Using a qualitative research methodology, specifically Grounded Theory, the research will systematically gather data regarding various issues in distributed agile projects. We have sought and have been granted approval by the University’s Human Ethics Committee to conduct interviews, surveys and observations.

The researcher aims to gather information by interviewing project managers, software developers and team leaders, agile practitioners and consultants who are involved with agile project management. An interview agenda has been attached to this information sheet outlining what data will be sought during the interviews. No personal details will be collected.

Purpose of Data Collection

The data gathered from participants will be analysed to derive understanding of critical and subtle issues of distributed agile projects that will help to manage future distributed agile projects successfully. Research papers may be published in conferences and journals for the benefit of the larger research community. The final PhD thesis will be published and will be held in the library at Victoria University of Wellington, New Zealand.
Attachment C: Consent for Research Participation

Research Topic: Distributed Agile Projects
Researcher: Siva Kumar Dorairaj

I have been given and have understood an explanation of the research study and the confidentiality conditions. I have had an opportunity to ask questions and have them answered to my satisfaction.

I agree to be interviewed by Siva Kumar Dorairaj for the purpose of this research which contributes to his PhD degree and resultant thesis and publications. I also understand that I may withdraw from this research up to one month after the interview.

I hereby consent to the collection and use of my opinions, perceptions, information and experiences during this research study.

Do you agree to have the interviews sound-recorded?

☐ Yes ☐ No

Would you like to receive a copy of any publications that are based on these interviews?

☐ Yes ☐ No

If Yes, please provide the e-mail or mailing address that can be used to send a copy of the publication.

___________________________________________________
___________________________________________________
___________________________________________________

Name: ______________________________
Signed: ______________________________
Dated: ______________________________
Attachment D: Interview Guide

General Information

Interview Date: ___________________
Interview Venue: _________________________________________________________
_____________________________________________________________________

Topic: Distributed Agile Projects

We may discuss any or all of the following depending on what is relevant and applicable to the interviewee’s experience in the distributed agile projects.
- How to manage distributed agile projects?
- How to disseminate progress and activity information to ensure a high level of communication and active interaction amongst all stakeholders are maintained?
- What are significant best practices and problems encountered during the projects?

Agenda:

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th>Duration (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Explain the topic, agenda and rules of this interview.</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>Experience with agile methodologies, agile project management and distributed agile projects.</td>
<td>10</td>
</tr>
<tr>
<td>3.</td>
<td>Define the role and responsibilities in the distributed agile projects.</td>
<td>10</td>
</tr>
<tr>
<td>4.</td>
<td>Explore the best practices in the distributed agile projects that contributed significantly to the success of the project.</td>
<td>15</td>
</tr>
<tr>
<td>5.</td>
<td>Discuss issues and difficulties the team encountered in the distributed agile projects.</td>
<td>10</td>
</tr>
<tr>
<td>6.</td>
<td>Provide recommendations for improvements based on user-experiences or scientific findings.</td>
<td>5</td>
</tr>
<tr>
<td>7.</td>
<td>Conclude (follow-up session and feedback)</td>
<td>5</td>
</tr>
</tbody>
</table>

Rules of Interview:
- Interviews will be sound-recorded, upon mutual agreement.
- Interviewees can be provided with interpretation of their comments during the interview, if required by the interviewee.
- Interviewee may discuss any other relevant issue not covered by the interview agenda.
Interview Questions.

1) Are you currently involved in a distributed Agile project?
   - If not, when was your last involvement in a distributed Agile project?
   - If yes, what is the current status of the project?

2) What is your role and responsibilities in the project?

3) What are the Agile methods being used in the project?

4) What is the size of the team?

5) How is the team distributed?

6) What is the duration of the project?

7) Do you face any challenges on this project?
   - If yes, what are those challenges?
   - What are the main causes of those challenges?

8) Are there any strategies that you adopted to overcome these challenges?
   - If yes, what are those strategies?

9) Are there any critical success factors in this project?
   - If yes, what are those critical success factors?

10) Is there anything else that we should have discussed?
    - If yes, please elaborate.
Appendix D: Coding Structure.

Overview of Coding Structure.

Strategy: Same Team.
Strategy: Same Time.
Strategy: Same Space.
Strategy: Same Culture.
Strategy: Same Practices.

Factor: Trust.
Factor: Senior Management Support.

Agility in Context
Appendix D provides the coding structure for the emergent theory (focusing on the core category: “becoming one team”). It comprises substantives codes, concepts, and categories that emerged from the data during data analysis. This coding structure supplements the detailed data analysis process described in section 3.7.1.

Step by step data analysis process (substantive coding):

1) We identified key points from the data (interviews transcripts and observation field notes).
2) We assigned a **code** to each key point in order to begin the data analysis process (open coding).
3) Using constant comparison method, each code was constantly compared with one another. We grouped together the codes that were related to a common theme in order to produce a higher level of abstraction: **concept**.
4) We continued to use constant comparison method on concepts, often consolidating similar concepts and renaming them. We grouped together the concepts that were related to a common theme in order to produce another higher level of abstraction: **category**.
5) The data analysis process gave rise to a large number of codes, concepts and categories. In this coding structure, however, we present the codes, concepts and categories that relate to the **core category** because the generation of a theory occurs around the core category.

Legend:
Same Team

Knowing the Team
- A face for the name
- Photographs of team members
- Recognise one another
- Sense of togetherness
- Simple sketch
- Understand team structure
- Wall of Fame

Engaging with the Team
- Active participation during meetings
- Beyond work hours
- Customer collaboration
- Organising outing with visitors
- Share personal information
- Suitable tools for communication
- Willingness to engage with team

Establishing Shared Vision
- Aligning with project goals
- Consistent and complete vision
- Importance of shared vision
- Knowing project vision
- Realising common goals
- Realising value for customer
- Shared understanding of project
- Understanding strength and weakness
- Using tools for sharing vision

Team Recognition
- Evaluate individual performance
- Recognise achievements of team
- Common rewards system across sites
- Support, not blame
- Paradigm shift for managers
- Politics within a team
- Complement, not compete
Same Time

Planned Formal Meetings
- Encourage full participation
- Extra effort for communication
- Fast feedback from customer
- Opportunity for interaction
- Promote knowledge acquiring
- Purposeful formal meeting
- Scrum ceremonies
- Sharing technical expertise
- Understand benefits of meetings
- Visibility into project status

Planned Unconference
- Allocated time for interaction
- Casual interaction in meetings
- Catch-up after meeting
- Meeting before meeting
- Scheduled casual chats
- Short but necessary interaction
- Unplanned meeting

Spontaneous Conversation
- Self-initiated conversations
- Casual, unplanned chats
- Importance of casual conversation
- Spontaneous need to communicate
- Conversation without purpose
- Unplanned meeting

Shifting Working Hours
- Flexible work hours
- Extended work hours
- Creating adequate time overlap
- Stretching work hours
- Realising importance of time overlap

Real-time Collaboration
- Tools support collaboration
- Importance of collaboration
- Fast feedback loops
- Effective collaboration
- Collaboration across sites
- Collaboration with customer
- Distributed pair-programming
- Instant response from team
Same Space

Inception Workshops
- Benefits of inception meetings
- Comprehensive discussion
- First communal meeting
- Initial meeting
- Interactive structured workshops
- Project feasibility meeting

Cross-site Visits
- Customer travels
- Senior members visit
- Stakeholders visit
- Subject matter experts visit
- Team ambassadors
- Travel to remote sites

Rotation
- Same role swap sites
- Maintain team structure
- Experience at remote site

Coach Travels
- Coach a coach
- Coach cultivates togetherness
- Coach improves interaction
- Conduct effective coaching
- Consistent coaching across sites
- Establish personal connection
- Importance of coaching

Temporary Co-location
- Co-locate with customer
- Co-located team
- Rotate location for co-location
- Start co-located
- Upclose and personal
- Whole team together
Agility in Context

Time Overlap
- Adequate time overlap
- Create time overlap
- Ideal time overlap
- No time overlap
- Similar time zone

Number of Sites
- Extreme number of sites
- Ideal number of sites
- Preferred sites
- Several sites within country
- Team availability

Team Size
- Extra travel cost
- Ideal size
- Little speaking time
- Travel decision
- Typical team size

Technical Infrastructure
- Choice for tools
- Efficient technological tools
- Improper infrastructure
- Leverage communication tools
- Reliable technology
- Supportive infrastructure
- Technological availability

Project Criticality
- Complex situation
- Critical circumstances
- Customer inavailability
- Ordinary situations
- Major situations
- High complexity
- Loss of business
- Customer satisfaction
Trust

Social Communication
- Casual talks before meetings
- Chatting using IRC
- Interaction over phone
- Outing with cohorts
- Personal conversation
- Purposeful conversation
- Random conversation
- Respectful conversation

Evidence of Trust
- Exhibit expertise
- Knowledge sharing
- Known technical skills
- Offer technical support
- Share business knowledge
- Share experiences
- Voluntary technical sessions

Collaboration Initiatives
- Collaboration across sites
- Customer collaboration
- Engaging work habits
- Express to collaborate
- Focus on project progress
- Passionate to work together
- Respect team work

Regular Communication
- Direct interaction between members
- Emphasise verbal communication
- Engaging customers
- Frequent communication
- Progressive communication
- Promote communication

Substantive Feedback
- Comprehensive response
- Effective feedback loops
- Honest feedback
- Reflective feedback
- Supportive environment
- Transparent responses

Team above Self
- Compromise differences
- Supportive individuals
- Team benefit
- Team interest
- Understand collective goals
- Willing to accept changes

Communication with Concern
- Understand inconvenience
- Realise the difficulties
- Show caringness
- Speak with concern
- Avoid confrontation
- Knowing the constraints
Bibliography


[13] AMBLER, S. W. IBM agility@ scale: Become as Agile as you can be. IBM, E-Book (2010).


315


317


324


[158] Hussain, Z., Slany, W., and Holzinger, A. Investigating Agile user-centered design in practice: A grounded theory perspective. In *HCI and


[211] MILLER, G. The magical number seven, plus or minus two: Some limits on our capacity for processing information. The psychological review 63 (1956), 81–97.


[256] Sabherwal, R. The role of trust in outsourced IS development projects. *Communications of ACM* 42 (February 1999), 80–86.


[291] SUTHERLAND, J., SCHOONHEIM, G., KUMAR, N., PANDEY, V., AND VISHAL, S. Fully distributed scrum: Linear scalability of production be-


[301] **Tuckman, B. W.** Developmental sequence in small groups. *Interpersonal growth and self-actualization in groups* 63, 6 (1965), 384–399.


