Global Software Development: A Multiple-Case Study of the Realisation of the Benefits

Eoin Ó Conchuír

Supervised by:
Professor Brian Fitzgerald

Co-supervised by:
Professor Helena Holmström Olsson

A Dissertation submitted for the Degree of Doctor of Philosophy at the University of Limerick

Submitted to the University of Limerick, May 2010
Abstract

Global software development (GSD) encompasses all software development where people located in different parts of the world are working together to produce software. Software development in itself is a complex process. GSD increases those complexities, as it involves people coordinating across significant geographical and temporal distance. It also typically involves people working together who come from different national and organisational cultures.

This mixture of distances brings with it both challenges and benefits. While working across significant distances can prove to be problematic, there have also been numerous benefits associated with GSD. Through the medium of Internet communication technologies, companies can tap into new expanding employment markets offering thousands of graduates with good education in software development skills. Moreover, these expanding employment markets typically demand salaries that may be many times lower than at higher-cost locations such as the US and Western Europe.

This thesis focuses on two multi-national companies with significant GSD activities. The aim of the study was to contribute to the body of knowledge regarding the potential benefits of GSD. Previous studies have offered some conflicting conclusions on characteristics of GSD – whether those characteristics are indeed beneficial and whether they are being realised in industry. While providing a synthesis and characterisation of numerous potential benefits of GSD through a structured framework, the thesis also examines in detail the realisation of these benefits in practice. The conclusions of the empirical study will help both researchers and practitioners to better understand which benefits are already being realised in practice, and how they can best be realised.
Declaration

The work presented in this thesis is the original work of the author, under the supervision of Professor Brian Fitzgerald. Where other sources of information have been used, they have been acknowledged. No part of this thesis has been previously submitted to this or any other university.

_________________________
Eoin Ó Conchúir
**Contents**

Abstract ................................................................................................................................................... i  
Declaration ............................................................................................................................................ ii  
Contents ................................................................................................................................................ iii  
Index of Figures ................................................................................................................................... x  
Index of Tables .................................................................................................................................... xi  
Non-Disclosure of Confidential Data ............................................................................................. xii  
Further Information ......................................................................................................................... xiii  
Acknowledgments ............................................................................................................................. xiv  

Chapter 1. Introduction ....................................................................................................................... 1  
  1.1 Background of the study ........................................................................................................ 1  
  1.2 Definition of key terms .......................................................................................................... 1  
  1.3 The rise of GSD ...................................................................................................................... 4  
  1.4 The benefits of GSD .............................................................................................................. 5  
  1.5 Research objective and questions ......................................................................................... 7  
  1.6 Thesis structure ....................................................................................................................... 8  

Chapter 2. Literature review ............................................................................................................... 9  
  2.1 Introduction ............................................................................................................................. 9  
  2.2 Co-located software development context .......................................................................... 9  
  2.3 Underlying processes of software development .................................................................. 10  
    2.3.1 Communication ........................................................................................................... 10  
    2.3.2 Coordination ................................................................................................................ 11  
    2.3.3 Control .......................................................................................................................... 12  
  2.4 Global Software Development ........................................................................................... 13  
    2.4.1 Geographical distance ................................................................................................ 13  
    2.4.2 Temporal distance ....................................................................................................... 14  
    2.4.3 Socio-cultural distance ................................................................................................ 15  
    2.4.4 The effect of the distances on software development ........................................... 16  
  2.5 Communication and Geographical Distance .................................................................... 18  
    2.5.1 Proximity to market/customer ................................................................................. 18  
    2.5.2 Improved focus on documentation ........................................................................... 18
2.5.3 Lack of informal and face-to-face communication ............................................... 19
2.5.4 Increased effort to initiate contact ........................................................................... 20
2.5.5 Cost of travel ............................................................................................................... 21
2.5.6 Dependency on information and communication technologies ............................... 21
2.5.7 Providing technical infrastructure .......................................................................... 22

2.6 Communication and Temporal Distance ......................................................................... 23
2.6.1 Reliance on asynchronous communication technologies ..................................... 23
2.6.2 Delayed feedback ........................................................................................................ 23
2.6.3 Shifting work hours .................................................................................................... 24

2.7 Communication and Socio-Cultural Distance .................................................................. 25
2.7.1 Asynchronous communication preferred by non-native speakers ...................... 25
2.7.2 Language differences and misunderstandings ........................................................ 25
2.7.3 Cultural differences and misunderstandings ........................................................... 26
2.7.4 Managing frames of reference .................................................................................. 27

2.8 Coordination and Geographical Distance ........................................................................ 28
2.8.1 Access to large skilled labour pool ........................................................................... 28
2.8.2 Access to cost-effective labour pool ........................................................................ 28
2.8.3 Reduced trust .............................................................................................................. 29
2.8.4 Lack of awareness/team spirit .................................................................................. 30
2.8.5 Cross-site modularisation of work ........................................................................... 31
2.8.6 Lack of mechanisms for creating shared understanding ....................................... 32
2.8.7 Improved process definition ..................................................................................... 33
2.8.8 Increased effort in outsourcing arrangements ........................................................ 33
2.8.9 Perceived threat from low-cost alternatives ........................................................... 34

2.9 Coordination and Temporal Distance ............................................................................... 35
2.9.1 Reduced hours of collaboration .............................................................................. 35
2.9.2 Time zone effectiveness ............................................................................................ 35
2.9.3 Time zone efficiency .................................................................................................. 36
2.9.4 Synchronous team meetings difficult to arrange .................................................... 37
2.9.5 Availability of technical infrastructure ..................................................................... 37
2.9.6 Coordination complexity ........................................................................................... 37

2.10 Coordination and Socio-Cultural Distance ..................................................................... 38
2.10.1 Innovation and shared best practices ................................................................. 38
2.10.2 Language and cultural training ................................................................................ 39
2.10.3 Lack of domain knowledge ..................................................................................... 40
2.10.4 Doubtful of others’ capabilities ............................................................................. 40
2.11 Control and Geographical Distance

2.11.1 Allocation of roles and team structure

2.11.2 Lack of concurrent software engineering principles

2.11.3 Increased autonomy

2.11.4 Organisational risks

2.12 Control and Temporal Distance

2.13 Control and Socio-Cultural Distance

2.13.1 Adapting to local formalised norm structures

2.13.2 Different perceptions of authority/hierarchy

2.14 A framework of the challenges and opportunities in GSD

2.15 Researcher’s reflections

Chapter 3. The Potential Benefits of Global Software Development

3.1 Introduction

3.2 Potential benefits

3.3 Proximity to market and customer

3.4 Improved focus on documentation

3.5 Record kept of communications

3.6 Access to large skilled labour pool

3.7 Access to cost-effective labour pool

3.8 Cross-site modularisation of work

3.9 Improved process definition

3.10 Time zone effectiveness

3.11 Time zone efficiency

3.12 Innovation and shared best practice

3.13 Allocation of roles and team structure

3.14 Increased autonomy

3.15 Summary

Chapter 4. Research Methodology

4.1 Introduction

4.2 Interpretive study

4.2.1 Interpretivism – ontology and epistemology

4.2.2 Positivism

4.2.3 Critical

4.2.4 Discussion

4.3 Research Methodology
5.2.16 Emerging theme: Team structure adding to overheads ................................................. 137
5.2.17 Emerging theme: Costing of GSD projects ................................................................. 140

5.3 Case 2: Semicon .......................................................................................................................... 141
5.3.1 Organisational context ........................................................................................................ 141
5.3.2 Team structure .................................................................................................................... 144
5.3.3 Proximity to market/customer .......................................................................................... 148
5.3.4 Improved focus on documentation ................................................................................... 149
5.3.5 Record kept of communications ....................................................................................... 149
5.3.6 Access to large skilled labour pool .................................................................................. 150
5.3.7 Access to cost-effective labour pool ................................................................................. 151
5.3.8 Cross-site modularisation of work .................................................................................... 153
5.3.9 Improved process definition ............................................................................................. 155
5.3.10 Time zone effectiveness .................................................................................................. 155
5.3.11 Time zone efficiency ....................................................................................................... 156
5.3.12 Innovation and shared best practice .............................................................................. 156
5.3.13 Allocation of roles and team structure .......................................................................... 157
5.3.14 Increased autonomy ........................................................................................................ 157
5.3.15 Within-case analysis: Evidence and drawbacks of the benefits .................................. 158
5.3.16 Emerging theme: Team structure to reduce coordination overheads ......................... 160
5.3.17 Emerging theme: Costing of GSD projects ................................................................. 162
5.3.18 Conclusion of case findings ............................................................................................ 163

Chapter 6. Cross-Case Analysis .................................................................................................. 164

6.1 Introduction .............................................................................................................................. 164
6.2 Initial contrasts ....................................................................................................................... 164
6.3 A comparative analysis of the case findings......................................................................... 167
6.3.1 Proximity to market and customer .................................................................................. 169
6.3.2 Improved focus on documentation & improved process definition ............................ 170
6.3.3 Record kept of communications ....................................................................................... 171
6.3.4 Access to large skilled labour pool .................................................................................. 172
6.3.5 Access to cost-effective labour pool ............................................................................... 174
6.3.6 Cross-site modularisation of work .................................................................................... 175
6.3.7 Time zone effectiveness .................................................................................................. 179
6.3.8 Time zone efficiency ........................................................................................................ 180
6.3.9 Innovation and shared best practices .............................................................................. 181
6.3.10 Allocation of roles and team structure .......................................................................... 182
6.3.11 Increased autonomy for software development units .................................................. 182
Index of Figures

Figure 4-1. Summary of research implementation................................................................. 93
Figure 5-1. Illustrative example of Pennysoft's organisational structure, taking Payroll
   Interfaces as a sample project......................................................................................... 107
Figure 5-2. Pennysoft team structure across geographical distance............................... 111
Figure 5-3. Typical structure of inter-site communication within one team at Pennysoft. .. 113
Figure 5-4. Overlapping working hours between India, Ireland and the US, with forces
   acting on standard working hours. ................................................................. 126
Figure 5-5. Illustrative example of Semicon's software program structure................. 142
Figure 5-6. Semicon team structure. ................................................................. 145
Figure 5-7. Typical structure of inter-site communication between teams in Semicon. .... 147
Figure 6-1. Cross-case analysis of interplay of motivation to reduce costs with the
   distances, processes, and emerging themes................................................................. 166
Figure 6-2. Level of task allocation as perceived in Pennysoft........................................ 177
Figure 6-3. Level of task allocation as perceived in Semicon............................................ 177
Index of Tables

Table 1-1. The relationship between offshoring and outsourcing. Adapted from Holmström Olsson et al. (2008). .................................................................4
Table 2-1. The distances affecting the software development processes. ...............17
Table 2-2. Framework of challenges and opportunities of GSD. .............................45
Table 3-1. Potential benefits of GSD as identified by the GSD framework. ...............50
Table 4-1. Summary of research epistemologies. .........................................................69
Table 4-2. Popularity of research methods in IS. .........................................................72
Table 4-3. Issues with conducting qualitative interviews, and a summary of how they were addressed by the researcher. Adapted from Myers and Newman (2007). ..............................................................................................................85
Table 4-4. Summary of activities for collection of empirical data.................................98
Table 5-1. Pennysoft interviewee codes, projects, and roles. .......................................109
Table 5-2. Approximate rate per hour, as charged by each site to internal customers. 122
Table 5-3. The realisation of benefits of GSD at Pennysoft. ........................................133
Table 5-4. The effect of the site hierarchy on the realisation of the potential benefits, where applicable..............................................................................137
Table 5-5. The effect of the cross-site team structure on the realisation of the potential benefits, where applicable.........................................................139
Table 5-6. Semicon interviewee codes, projects, and roles. .........................................144
Table 5-7. Approximate rate per hour, as charged by each site to internal customers. Assume x to be the per-quarter cost charged by India. .........................152
Table 5-8. The realisation of the benefits of GSD at Semicon. ....................................159
Table 5-9. The effect of the team structure on the realisation of the potential benefits, where applicable..............................................................................162
Table 6-1. Summary of within-case findings from both cases. ...................................168
Table 6-2. A distribution table of the realisation of the benefits of GSD. .....................183
Table 7-1. Proposed classification of the potential benefits of GSD.............................198
Non-Disclosure of Confidential Data

The selected excerpts taken from interview transcripts which are included in this thesis are used by permission of the interviewees on the understanding that they will not be disclosed outside this dissertation. In deference to this confidentiality agreement, readers of this dissertation are not permitted to reproduce these excerpts in any form, written or oral, or use them as sources of secondary data, without the direct consent of the author.
Further Information

Readers of this dissertation are welcome to contact the author at the following e-mail address:

Eoin Ó Conchuir <eoinoconchuir@gmail.com>
Acknowledgments

I would like to express my sincere gratitude to all those (named or not) who helped me complete this part of the road of life.

A special thanks goes to Professor Brian Fitzgerald who supervised this project, and to Professor Helena Holmström Olsson who co-supervised. They offered countless hours of advice, support, and guidance. I thank them both for their commitment throughout the ups and downs of my completing this dissertation. Similar thanks goes to my colleagues at Lero for their input.

I would also like to thank the examiners, Dr. Brian Nicholson and Dr. Gabriela Avram, for their helpful comments and suggestions.

My gratitude goes to those company representatives who allowed for this empirical study to take place and to the interviewees themselves for their time and insights.

I dedicate this work to my wife and best friend Saša for her kind and unending support.

This project was partly funded under the B4Step project (Grant no. SFI 02/IN.1/108), and by Lero – The Irish Software Engineering Research Centre.
Chapter 1. Introduction

1.1 Background of the study

Global software development (GSD) arises when the people coordinating to develop a software system are typically based in different countries, across different time zones and coming from different cultural backgrounds. GSD brings with it a variety of potential benefits – most notably the potential to reduce costs while tapping in to large multi-skilled labour pools. GSD also carries with it many challenges in dealing with the global distances that divide those coordinating the system development. GSD is becoming common practice in the software industry (Damian and Dustdar 2005) and is rapidly becoming the norm for large software systems (Herbsleb 2007). This thesis presents an exploratory study of the potential benefits of GSD. This chapter serves to set the context of the study. Firstly, key terms are defined. Secondly, an introduction of the rise of GSD is presented, followed by an overview of the potential benefits of GSD. Thirdly, the research objectives and research questions that drove the study are presented.

1.2 Definition of key terms

Software Development

For the purpose of this thesis, *software development* is broadly defined as “any software development lifecycle activity” (Ågerfalk et al. 2005). This extends beyond ‘pure’ development activities and includes, for example, deployment and maintenance phases. The term “activity” is applied in a broad sense, including any individual or collective human action at any level of granularity that serves a particular purpose. According to activity theorist Engeström (2000), an activity is something that transforms an object to an outcome. Hence, a “development activity” is that individual or collective action that transforms something (abstract or concrete) into something meaningful in the context of a software system’s lifecycle. Thus, an individual developer’s creation of a source code document is a development activity of transforming a requirements document into a piece of code. The individual carrying out the activity is the actor. An “actor” will be used synonymously with “team member”, as it is the team members that act during a software development project to produce the resulting product. The development of a project may
Global Software Development

The term *global software development* has been used by many, (e.g. Mockus and Herbsleb 2001, Bass and Paulish 2004, Damian and Lanubile 2004, Herbsleb et al. 2005, Abrahamsson et al. 2008). The term is often used without precise definition. Ebert and De Neve (2001) define the term by giving the example of opening a software development centre in another region of the world. Vanzin et al. (2005) and Mockus and Herbsleb (2001) define it to be software development activities distributed globally across several sites that are located in different countries and continents.

The term “global software development” unfortunately carries with it ambiguities. The usage of the term is not meant to focus on the development of “global software”, but rather describes the act of developing software systems in a global setting. Thus, it describes software development efforts that include actors who are spread out globally, and not only in one location or one country. The global distances may or may not define the boundaries of teams. For example, GSD equally describes a virtual team where the team members are dispersed globally, or a project involving teams with co-located team members but with each team located at a separate location. The term “distributed software development” has also been used (Kiel 2003, Damian and Dustdar 2005), describing the act of developing software in geographically distributed settings, but not necessarily on a global scale. Another suitable term which carries with it less ambiguity is “globally-distributed software development” (Carmel 1999, Herbsleb and Mockus 2003, Boland and Fitzgerald 2004). The term “global software engineering” also came to prominence in 2006 with the first International Conference on Global Software Engineering. For the purpose of this thesis, the following definition of the term Global Software Development (GSD) is adopted, similar to that defined by Ågerfalk et al. (2005):

> All activities of the software lifecycle where the project involves actors who are dispersed across at least two locations which are separated by country or continental borders, and typically across multiple time zones with a degree of socio-cultural distance amongst the actors.

Offshoring and Outsourcing

The terms are used in this thesis to define certain characteristics of software development efforts, and so must be defined here. The two terms describe two different phenomena,
but are not mutually exclusive in context. Offshoring is related to geography, while outsourcing is a governance term. At one point, the term ‘offshore’ was financial, relating to ‘offshore’ bank accounts such as in the Cayman Islands.

Offshoring has now come to mean the completion of tasks (both manufacturing and services) in lower-cost economies (Reh 2004, Carmel and Tija 2005). An offshore location can be any other location outside the organisation’s home country (Carmel and Tija 2005). The term is indeed US-centred, and has been used to describe significant shift of manufacturing jobs from the US to Asia, especially. More recently, offshoring has encompassed the offshoring of services, such as the offshoring of certain business processes. As defined by Venkatraman (2004), it is “the practice among U.S. and European companies of migrating business processes overseas to India, the Philippines, Ireland, China and elsewhere to lower costs without significantly sacrificing quality”. In the context of offshoring in GSD, some or all software development activities are completed outside of the company’s home country, typically in a lower-cost location. Low-cost locations would typically be those falling into the economic grouping of ‘developing nations’ or ‘emerging nations’, such as India, China and Russia, Brazil and Romania (Nicholson and Sahay 2001, Holmström Olsson et al. 2008). The term offshoring does not distinguish between tasks completed by an offshore division of the same company, or by an offshore, contracted, external company.

Outsourcing is related to governance rather than geography (Holmström Olsson et al. 2008). Outsourcing means that tasks and processes are contracted to be performed outside the boundaries of the organisation (Holmström Olsson et al. 2008). Outsourcing is the shedding of tasks by companies of even their core functions “by hiring them out to companies that [can] do them more efficiently and, thus, less expensively” (Reh 2004). There are a number of definitions describing outsourcing involving information systems (IS). Choen, Grover et al. (1995, p. 209) define IS outsourcing as “the organizational decision to turn part or all of an organization’s IS functions to external service provider(s) in order for an organization to be able to achieve its goals”. Likewise, Goles and Chin (2005, p. 49) define outsourcing as “…contracting with one or more third party vendors for the provision of some or all of an organization’s IS functions, where ‘functions’ include one or more IT activities, processes, or services to be provided over time”. Klepper (1995, p. 249) emphasises that outsourcing is between multiple entities: “Outsourcing is defined as the provision of services by a vendor firm to a client”.

An activity can be completed either onshore or offshore, and internally in the organization (in-house) or be outsourced to an external organisation. Software development that is
outsourced to an offshore vendor has been termed global software outsourcing (GSO) (Imsland et al. 2003). In that paper, Imsland et al. define it as “relatively long term relationship between firms based in different countries to enable software development to be carried out primarily off-shore (in the premises of the firm doing the development)”. The offshoring of work regardless of it being in-house or outsourced has been termed “offshore sourcing” (Carmel and Tija 2005). Finally, Table 1-1 below shows the relation between offshoring and outsourcing previously defined.

Table 1-1. The relationship between offshoring and outsourcing. Adapted from Holmström Olsson et al. (2008).

<table>
<thead>
<tr>
<th></th>
<th>In-house</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Onshore</strong></td>
<td>In-house (traditional model)</td>
<td>Subcontractor (third party) in the same locale</td>
</tr>
<tr>
<td><strong>Offshore</strong></td>
<td>Foreign branch of the same company (captive centre)</td>
<td>Subcontractor (third party) in a foreign locale</td>
</tr>
</tbody>
</table>

1.3 The rise of GSD

While manufacturing industries have been offshoring manufacturing processes to lower-cost destinations for thirty years or more, it wasn’t until the late 1980s that a significant portion of software development work was being sent offshore (Sahay et al. 2003). From that point onwards, India rose to become a dominant player in GSD, offering a large skilled workforce with proficient English language skills. By 1997, India was already exporting US$1 billion in software and software services annually, and by 1999 the country had roughly 200,000 well-trained software professionals (Carmel 1999).

Before the rise of GSD, multi-national companies had already realised that costs could be reduced by locating manufacturing in lower-cost countries. For example, Ó Riain (1997) cited by Sahay et al. (2003) traces Ireland’s hosting of multi-national companies back to the 1960s when companies such as Digital began manufacturing high-tech products there. This trend was enabled by favourable tax incentives and grants offered to the multi-nationals. The software industry followed the lead of the manufacturing industry by beginning to set up development activities in newly-developing countries in order to capitalise on cost savings. Several historical factors increased the trend of distributing software development activities (Lewin and Peeters 2006). Much maintenance work relating to the ‘millennium bug’ was sent offshore, introducing companies to the possibility of offshoring some of
their software maintenance work. As the IT bubble burst after 2000, companies sought to maximise cost savings, which led them to offshoring more development work to cheaper labour markets.

As well as being the result of a drive to lower costs, GSD can be the side-effect of mergers, acquisitions and alliances (Carmel 1999, Grinter et al. 1999, Herbsleb and Moitra 2001). After a strategic merger or acquisition to expand product families (Carmel 1999, p. 5), a company may find that it has software development expertise in several locations. Such a context may result in teams that originated from separate companies coordinating globally to develop software. Governmental policies may also encourage an organisation to establish software development activities in that country. In some markets government regulations request the location of some local development operations (Herbsleb et al. 2000).

GSD is now commonplace in software development companies (Herbsleb 2007). With increasing globalisation of markets and production, there is increasing pressure to distribute software development globally. As such, GSD will continue to be a phenomenon of great importance into foreseeable future (King and Torkzadeh 2008). Countries such as India will remain to be attractive destinations for offshoring IT services over the coming years (Kotlarsky and Oshri 2008).

1.4 The benefits of GSD

Over the past decade and more, companies have been able to leverage the emergence of large multi-skilled labour forces in lower-cost economies thanks to high-speed Internet-based communication links, through which the product (software code and other digital documents) can be quickly transferred between development sites (Herbsleb et al. 2000, Carmel and Agarwal 2001, St.Amant 2005). GSD is more than simply a side effect of mergers and acquisitions; it has been promoted as a beneficial approach for companies to develop their software. Below is a brief introduction to some of the most widely cited potential benefits of GSD:

- **Access to cost-effective labour pool.** The main enabler, and attraction, of GSD is the availability of a relatively lower-cost labour force compared to the costs in the organisation’s home country (Carmel and Agarwal 2001, Herbsleb and Moitra 2001, Damian et al. 2003). For example, in 2005 a professional IT engineer in China was earning an average annual salary of US$4,800 (EIU eBusinessForum.com 2005). In
the same year, the average annual base salary of a software development engineer in India stood at US$10,300 (Mercer 2005). In contrast, the median annual salary of a US computer applications software engineer in 2004 was nearly US$75,000 (Bureau of Labor Statistics - U.S. Department of Labor 2005). The up-front potential for huge savings in salaries has been the main drive behind GSD.

- **Access to large skilled labour pool.** GSD provides the unprecedented possibility to leverage large pools of skilled labour by coordinating across distance (Grinter et al. 1999, Carmel and Agarwal 2001, Ebert and De Neve 2001, Herbsleb and Moitra 2001, Damian et al. 2003, Herbsleb 2007). Companies have the opportunity to expand their software development activities to include the contributions of thousands of skilled workers, wherever they may be located, to quickly form virtual global corporations (Suzuki and Yamamoto 1999, Herbsleb and Moitra 2001, Carmel and Tija 2005).

- **Proximity to market and customer.** By establishing subsidiaries in countries where the company’s customers are located, GSD allows it to develop software close to its customers and to increase knowledge of the local market (Grinter et al. 1999, Herbsleb et al. 2000, Herbsleb and Moitra 2001, Damian et al. 2003). Creating new jobs can also enhance good will with local governmental customers, possibly resulting in more contracts (Ebert et al. 2001).

- **Time zone effectiveness.** Time zone effectiveness is the degree to which an organisation manages resources in multiple time zones, maximising productivity by increasing the number of hours during a 24-hour day that software is being developed by its teams. The approach can help companies reduce their time-to-market when under severe time pressure (Herbsleb and Moitra 2001). When time zone effectiveness is maximised, development spans 24 hours of the day. This has been termed *follow-the-sun* development (Carmel 1999).

- **Innovation and shared best practice.** Organisations can take advantage of increased innovation and shared best practice that arises from the collaboration of team members who come from different national and organisational backgrounds (Ebert and De Neve 2001, p. 10, Carmel and Tija 2005).

The potential benefits of GSD have sometimes been questioned. Despite the advantages of GSD, Ebert and De Neve (2001) claim that “global software development is not the target per se,
but rather the result of a conscious business-oriented trade-off”. In fact, the cost-benefit trade-offs for GSD are complex and not well understood (Espinosa and Carmel 2004). There are many challenges to achieving efficient GSD within increasingly virtual organisational structures, with global distances affecting the processes of communication, coordination, and control between actors (Damian and Zowghi 2002, Damian et al. 2003). Software development itself is a complex undertaking, and separating actors by global distances make the task more difficult. The global distances between actors leads to an increase in coordination costs, which is the effort required to work together (Espinosa and Carmel 2004). Carmel and Agarwal (2001) go so far as to suggest that GSD should be avoided if possible because of the complexities involved. Meanwhile, Kobitzsch et al. (2001) claimed that there may be a growing consensus that cost savings may not be the only potential benefit of GSD, but rather that there are other more compelling reasons to globally source IT skills.

1.5 Research objective and questions

There is a void in the understanding of the potential benefits of GSD, and how they interplay with the difficulties involved with the inherent global distances. On the one hand, previous literature has pointed to the attractive benefits that GSD can offer, but on the other hand distributed software development is seemingly a complex undertaking that perhaps should be avoided. There is a lack of understanding of the intricacies of each of the potential benefits, and how they are being realised in practice. It is not clear if the benefits are being realised, or how their effect should be maximised. It is these gaps in knowledge that this thesis seeks to address.

The objective of the research study described in this thesis was to provide an in-depth investigation of this area with two specific areas of focus:

- **RQ1**: Identification and framework classification of potential benefits of GSD
- **RQ2**: Investigation of the extent to which these potential benefits are being realised in practice, and the underlying reasons for realisation/non-realisation
1.6 Thesis structure

This chapter has laid out the context of GSD and the research questions that drove the study described in this thesis. The document is structured according to the following chapters:

- Chapter 2 presents a critical review of related literature, structured through a framework of challenges and opportunities of GSD. The opportunities identified in the chapter are taken to be the potential benefits of GSD. The framework is built upon in later chapters to help analyse the research data of the study.

- Chapter 3 further characterises each potential benefit according to existing literature. The characterisations are later used to compare to the findings of the empirical study.

- Chapter 4 describes the epistemological and ontological positioning of this research study, the research method adopted to collect and analyse the data. The research sites are also presented.

- Chapter 5 presents the empirical findings of each of the two cases of the study in isolation, followed by a within-case analysis that identifies emerging themes coming from that case.

- Chapter 6 draws together the within-case analysis of the previous chapter in order to compare and contrast the cases, providing a deeper understanding of the phenomenon.

- Chapter 7 discusses findings and analysis of the previous chapters in order to distil the contributing knowledge coming from the study.

- Chapter 8 concludes the thesis by reviewing the approach taken to performing the empirical study, and discusses both the limitations and contributions of the work.
Chapter 2. Literature review

2.1 Introduction

Software development in itself is a complex process. GSD inherits this complexity, and brings with it a strong accentuation of several distances – or degrees of distribution – amongst teams and team members. In this chapter a critical overview of the academic body of knowledge on GSD is presented. Specifically, underlying processes of software development are discussed (namely, communication, coordination, and control). Then, the global distances involved in GSD are identified (namely, geographical, temporal and socio-cultural distance). In order to characterise the opportunities and challenges of GSD, a critical review of existing literature is offered by examining how the primary processes of software development are affected by the global distances. As such, both the opportunities and challenges of GSD are presented as a framework according to the processes and distances identified. It is taken that the opportunities presented in this framework are synonymous with the benefits of GSD. While helping us to understand the phenomenon of GSD, this chapter shows that the GSD community has not settled upon a common understanding of the interplay between the benefits and challenges.

2.2 Co-located software development context

Software development team members must communicate whenever necessary to make the team efficient (DeMarco and Lister 1999). Co-location allows for an almost immediate response to a request for information or clarification, and also encourages unplanned informal communication which is deemed to be an important success factor for software development teams (Herbsleb 2007). Face-to-face communication offers “richer” communication (Carmel 1999, p. 41), compared to alternative settings. Therefore, the ideal environment for a software development team is for all team members to be co-located at one site, with collaboration within the team and control of the team facilitated by frequent face-to-face contact.
2.3 Underlying processes of software development

An organisational unit (such as a team, or project) may execute complex organisational processes – in this case, the act of developing software. Software development is “truly a human process of interpretation, interaction and reality construction” (Rönkkö 2007). An organisational unit cannot function coherently without the processes of coordination (Malone and Crowston 1994) and control (Carmel and Agarwal 2001).

Coordination is “the management of dependencies among task activities to achieve a goal” (Espinosa and Carmel 2004). This process often requires intense and on-going communication (Carmel and Agarwal 2001). Control involves the structures required ensuring development of software in time, on budget and of desired quality. Communication is the mediating factor for both coordination and control in software development, as it allows for information to be exchanged. Below, each of the three processes are outlined.

2.3.1 Communication

Communication is “the action of communicating heat, feeling, motion, etc; specifically the transmission or exchange of information, news, etc.” (Oxford University Press 2002). Communication is “the exchange of complete and unambiguous information – that is, the sender and receiver can reach a common understanding” (Carmel and Agarwal 2001). The communication process concerns the transfer of knowledge and information between actors, and the tools used to facilitate such interaction (Ågerfalk et al. 2005).

The most explicit communication medium between humans is language, and this is enriched, for example, by facial expressions, hand movements and tone of voice. Non-verbal indicators that support face-to-face communication make it easier for the receiver to interpret the message’s intent and for the sender to ascertain that the message was received and understood (Karolak 1998, pp. 63-72). The language of face-to-face conversation is the basic and primary use of language, all others being best described in terms of their manner of deviation from that base (Fillmore 1981). Karolak (1998) defines the effectiveness of a communication method by its timeliness and content. Timeliness is how quickly a communication is received. Content is the amount of verbal or written communication, and what can be read into it, such as through facial expressions or tone of voice.

Communication is an essential process within all software development activities (Curtis et al. 1988, p. 1278, Perry et al. 1994). The processes of coordination and control within GSD are dependent upon the process of communication (DeLone et al. 2005). For the purposes
of this thesis, ‘communication’ includes both formal and informal communication during all stages of software development. Formal communication in that context includes formal requirements specifications documents, a modification request dealt with in a change management system, scheduled meetings, test plans and delivery schedules (Kraut and Streeter 1995, Herbsleb and Mockus 2003, Damian and Dustdar 2005). Informal communication includes all face-to-face communication outside of meetings, such as “coffee chats” (Damian and Zowghi 2002), unplanned corridor chats, talking to a person at their desk to resolve a problem, sending an instant message and sending an e-mail with a request for clarification. This can account for 75 minutes of each working day (Perry et al. 1998). Informal contact channels allows developers to fill in details, to handle exceptions and to correct mistakes (Herbsleb and Grinter 1999b).

In fact, unplanned informal contact is a huge part of co-located software development activities (Curtis et al. 1988, Beck 2000), the lack of which is discussed later. Informal contact allows team members to develop working relationships, and allows a better flow of information about changes in the current project (Herbsleb and Mockus 2003). Both formal and informal communication are implicit in allowing a team to build a shared working language, and establishing visibility of with whom expertise lies (Herbsleb 2007). Informal contact is especially important in unstable dynamic teams (DeLone et al. 2005). Handling the unanticipated both rapidly and gracefully requires flexible ad hoc communication (Herbsleb and Grinter 1999a). Kraut and Streeter (1995) observed in a survey of 65 projects in one large development company that as the size and complexity of the software increases, the need for supporting informal communications increases dramatically. This shows that communication is used or even required in the face of added complexity. In effect, communication allows for the people involved in developing the software to coordinate their efforts. The following section discusses the act of coordination, or the management of dependencies.

2.3.2 Coordination

Coordination is “orderly combination; the action or result of placing things in due order or relation to each other; the harmonious functioning together of different interrelated parts” (Oxford University Press 2002). It is “the act of integrating each task with each organisational unit, so the unit contributes to the overall objective” (Carmel and Agarwal 2001). Espinosa and Carmel (2004) continue with the theme of coordination being an act upon sub-units involved in achieving the overall goal: “the management of dependencies among task activities to achieve a goal”. This is consistent with
coordination as defined by coordination theory (Malone and Crowston 1994). Kraut and Streeter (1995) view coordination as the activity played out by the sub-units themselves: “In software development, [coordination] means that different people working on a common project agree to a common definition of what they are building, share information, and mesh their activities”.

People coordinate to solve coordination problems (Clark 1996). Hence, the coordination process concerns how this interaction makes actors interdependent on each other. If two actors are working together on a task and their respective activities can be carried out independently, there is nothing to coordinate during their completion of the tasks (Espinosa and Carmel 2004). GSD would essentially not exist if there were no dependencies between remote actors. This makes coordination across distance the key phenomenon of GSD according to Herbsleb (2007). If their activities are highly interdependent, then the respective dependencies have to be effectively managed, often creating substantial coordination overhead.

Achieving effective coordination in its constituent parts is a major challenge for the software process (McChesney and Gallagher 2004). McChesney and Gallagher (2004) suggest three levels which have to be coordinated for effective software development. At an organisational level, a team can be structured to reduce interdependencies. At a process level, coordination can be applied through knowledge sharing and standardisation to effectively manage common procedures and techniques for software engineering. At a tools level, software engineering environments have provided technological support for coordinating complex system builds.

2.3.3 Control

Control is “the act or power of directing or regulating” (Oxford University Press 2002). In software development, it has been defined as “the process of adhering to goals, policies, standards, or quality levels” (Carmel and Agarwal 2001). The control process concerns the management and reporting mechanisms put in place to make sure a development activity is progressing. Control thus relates to project management. While control does differ from coordination (Sabherwal 2003), the processes are related (Nurmi et al. 2005). The act of coordinating team members, such as setting objectives and deadlines, places controls on their work activities.

Control can be exercised in software development by both formal and informal control mechanisms. Formal mechanisms may include setting goals, policies, standards, reports and
regular meetings. In a co-located setting, a manager exerts control through “management by walking around” (Carmel 1999) and other informal and nuanced control mechanisms. The mixture of formal and informal control mechanisms also change depending on the phase of the software development project.

2.4 Global Software Development

As GSD is software development in a distributed context, it inherits the processes, identified above, of communication, coordination and control (Carmel and Agarwal 2001, Evaristo et al. 2004, Nurmi et al. 2005). However, due to the global distribution involved, it also implies the distances of geographical distance, temporal distance and socio-cultural distance. The International Workshop on Global Software Development and, more recently, the International Conference on Global Software Engineering have highlighted the effect of the distances on distributed software development (Damian et al. 2003, Damian and Lanubile 2004, Damian and Dustdar 2005).

The term ‘global’ in GSD emphasises that the focus is on software development efforts that span separate countries, time zones, and even continents. The distributed aspect of GSD introduces perhaps the most evident distance – geographical distance on a global scale. Many core characteristics of GSD are related not only to geographical distance, but also related to what can be called “socio-cultural” distance. Socio-cultural distance describes the degree to which the cultural backgrounds of the team members differ, and this can affect their interpretation and response to a situation (Nicholson and Sahay 2001, Kiel 2003, Herbsleb et al. 2005). Another distance that can be implied by geographical distance is that of temporal distance, when distribution covers two or more time zones. Below, further contextual definitions of the distances are given. For this study, the three distances of geographical, temporal and socio-cultural distance are accepted to be the defining distances of GSD, as identified by Ågerfalk et al. (2005) and Carmel (1999).

2.4.1 Geographical distance

Rather than defining geographical distance by the linear distance between two team members or sites in kilometres, geographical distance is defined here as the directional measure of the effort required for one individual to visit another at the latter’s home site (Ågerfalk et al. 2005). Two locations within the same country with a direct air link and regular flights can be considered relatively close even if separated by great distance, but the
same cannot be said of two locations which are close due to the smaller number of kilometres between them but with little transport infrastructure and perhaps intervening borders. The latter case would imply a high geographical distance.

The ease of relocating has several factors, like the need to obtain visas, and the time it takes to complete the journey. In general, low geographical distance offers greater scope for periods of co-located, inter-team working, and for more unplanned informal communication (Herbsleb and Grinter 1999b). The geographical distance in GSD implies the need to relocate to other countries for face-to-face communication, with some distances spanning continents and requiring long-distance flights.

2.4.2 Temporal distance

Temporal distance is a directional measure of the dislocation in time experienced by two actors wishing to interact (Ågerfalk et al. 2005). This can be as a result of several factors, including being located in two different time zones (which is a result of geographical distance) and working hour norms for each site. In fact, temporal distance arises as soon as the hours of possible synchronous communication between two actors are reduced. In practical terms, temporal co-location occurs when both actors are working at the same (absolute) time for the duration of their workdays, allowing them to communicate synchronously at any point during their working day. Temporal distance increases as the number of overlapping work hours between the two actors is reduced.

Time zone differences and time shifting work patterns can work together to either increase or decrease temporal distance. For example, a one-hour difference in time zones can, due to different routines during a workday (e.g. different lunchtime norms), lead to few overlapping hours and an appearance of higher than expected temporal distance (Herbsleb and Grinter 1999b). Conversely, a European worker liaising with a counterpart in India who is working a late shift may experience low temporal distance.

Geographical distance and temporal distance are often correlated (Orlikowski 2002, Watson-Manheim et al. 2002, Espinosa et al. 2003b), making it harder to differentiate between effects from geographical distance and those that result from temporal distance based on current literature (Espinosa and Carmel 2004). Finally, large geographical distance does not necessarily imply temporal distance because the two locations may be within the same time zone.
2.4.3 Socio-cultural distance

Culture, at its most basic level, is a set of “shared symbols, norms, and values in a social collectivity such as a country” (Walsham 2002). Socio-cultural distance is a directional measure of an actor's understanding of another actor's values and normative practices (Ågerfalk et al. 2005). According to Carmel and Agarwal (2001), the distance is ‘difference’ between the actors involved in terms of national and organisational culture. This distance is not unique to GSD. Co-located teams may be comprised of team members coming from different national and organisational backgrounds.

*National culture*, or local culture, encompasses an ethnic group’s norms, values, spoken language, and styles of communication (Carmel and Agarwal 2001, Imsland et al. 2003). For example, norms for acceptable working hours can differ between countries (Robey et al. 2000). Very generally speaking, national cultural distance stems from the relative distance between the actors (Carmel and Agarwal 2001, p. 25). As local culture typically relates to the geographic position of its community, geographical distance can therefore imply a degree of socio-cultural distance. Socio-cultural distance may be experienced over a small geographical distance (for example, across country borders, or within a country with a diverse mixture of cultures). There may also be a low degree of socio-cultural distance across vast distances (such as across a large country with a homogenous national culture), but this can still present challenges. For example, Robey et al. (2000) found that a virtual team within the US experienced issues with accents and stereotyping.

*Organisational culture* encompasses the organisational unit’s norms and values, and this unit may range from a small external contractor to an enterprise-wide culture of a multi-national organisation (Carmel and Agarwal 2001). This has also been termed “corporate culture” (Herbsleb et al. 2005). Two actors coming from separate organisational cultures may have different views on authority, project management practices and dress code, for example (Krishna et al. 2004). Organisational norms also include practices of uses of programming methodologies (Carmel 1999, Carmel and Agarwal 2001), the speed with which it is normal to reply to an e-mail, and varying engineering styles (such as focusing more on the design stage or on the actual development stage) (Herbsleb et al. 2005). An organisational culture may include specific vocabulary and abbreviations that would not be readily understood by a person who has not been previously involved with the company. A related distance has been termed “organisational distance” (Espinosa et al. 2003a), that is the measures of density and centrality involving organisational affiliation. Organisational distance may
therefore increase where team members belong to different organisations, or where a team member also belongs to another team in parallel.

Cultural distance is an issue of high importance to researchers in the field, as found through a Delphi study described in the editorial of a special issue on global offshoring in the MIS Quarterly journal (King and Torkzadeh 2008). In the context of GSD, the different national and organisational culture of actors involved may lead to misunderstandings or conflicts (Casey and Richardson 2004). While the distance is not unique to GSD, the phenomenon can lead to a special circumstance where each team grouping has a coherent homogeneous local culture, but with a high degree of cultural separation from the other team groupings they must work with remotely.

2.4.4 The effect of the distances on software development

As with co-located software development, communication is an essential process of GSD (Carmel 1999, Beck 2000). The bandwidth of communication between members of a software development team is challenged by the distances of GSD due to the fact that GSD replaces the communication context of the ideal face-to-face setting (Fillmore 1981, Clark 1996) with a technology-mediated and thus more restricted one (Ågerfalk 2004). Geographical distance can directly affect the process of communication. Allen (1977) found that the level of communication was greatly reduced between colleagues who were separated by as little as thirty metres. Being located on two different levels of the same building can have a similar effect (Kraut et al. 1988). Distance exacerbates coordination and control problems directly, but also through its negative effects on communication (Carmel and Agarwal 2001). At the same time, when managing software development projects at diverse locations, communication methods and tools offer one of the most powerful and effective ways to gather and disseminate information and to control the project (Karolak 1998).

We know intuitively that it is harder to manage from a distance, thus affecting the process of control (Carmel 1999). According to Carmel (1999), while some mechanisms for the process of control within co-located teams are formal, much are informal. This can be as simple as a manager walking to a developer’s desk to check that sufficient progress is being made. Global geographical distance means that “management by walking around” cannot work for geographically-dispersed teams (Carmel 1999). Control is made even more difficult in non-routine, complex, and dynamic software development (Kirsch 2004).
Traditional control mechanisms may need to be adapted to allow managers to control team members who are situated at another location.

As has been indicated above, GSD places demands on the software process imposed by increased complexity related to communication, coordination and control. It is therefore worthwhile to frame the study of GSD according to how each process is affected by each distance. As such, this chapter examines GSD literature along the three broad categories of communication, coordination and control, with each category examined with respect to geographical, temporal and socio-cultural distance. See Table 2-1 for a tabular depiction of the categorisation:

Table 2-1. The distances affecting the software development processes.

<table>
<thead>
<tr>
<th>Process</th>
<th>Geographical</th>
<th>Temporal</th>
<th>Socio-Cultural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination</td>
<td>Lack of awareness; increases effort to coordinate.</td>
<td>Communicate asynchronously, or shift working hours.</td>
<td>Doubtful of others’ abilities.</td>
</tr>
<tr>
<td>Control</td>
<td>Cannot manage by walking around.</td>
<td>Asynchronous control over remote resources.</td>
<td>Cultural variations of control mechanisms.</td>
</tr>
</tbody>
</table>

Not all issues lie neatly into one cell of the table, but this presentation of information does allow us to approach the wide topic in a structured manner. While this section has briefly covered the potential negative effects of distance in GSD, there are also opportunities presented by the dispersed nature of GSD. As such, this chapter addresses both the opportunities and challenges of GSD. The following sections describe each cell of the table separately, and each cell is discussed in relation to its opportunities and challenges. The opportunities identified are further examined in subsequent chapters.

This thesis focuses on the area of GSD, and specifically on the software engineering aspects of working in a distributed setting when developing software. Offshore outsourcing is another perspective on this phenomenon, focusing in particular on developing software across organisational boundaries. While this thesis does not aim to address governance issues related specifically to outsourcing, some outsourcing studies are cited here. Such cited studies include Rottman and Lacity (2004) for their study on best practices for
offshoring, Lacity et al. (2009) for their review of IT outsourcing literature, and Imsland et al. (2003) for their insights into working across cultural distance.

2.5 Communication and Geographical Distance

2.5.1 Proximity to market/customer

By establishing subsidiaries in countries where the company’s customers are located, GSD allows it to develop software close to their customers and to increase knowledge of the local market (Grinner et al. 1999, Herbsleb et al. 2000, Herbsleb and Moitra 2001, Damian et al. 2003). Companies can therefore make use of distributed teams where part of the team is located in the target market (Herbsleb and Grinner 1999b). This allows for an increased knowledge of local customers and knowledge of local conditions (Herbsleb and Moitra 2001), and facilitates more effective requirements elicitation (Damian and Zowghi 2002). Furthermore, investing locally engenders goodwill in the host country (Ebert et al. 2001). Development activities may even be located on the same campus as the organisation’s large customer. Companies may look to establishing strategic partnerships to gain access to new markets (Karolak 1998), but governmental restrictions may also force the organisation to locate some of its development activities in the target market in order to be allowed to participate there (Herbsleb et al. 2000). This potential benefit may also be related to temporal distance if it is used to reduce the temporal distance with customers.

2.5.2 Improved focus on documentation

DeLone et al. (2005) state that distributed teams have an increased focus on documentation in order to aide their communication. Shared documentation may be required not only to overcome geographical distance, but also temporal and socio-cultural distance. However, written documentation is inadequate when resolving misunderstandings, such as misunderstandings about requirements (Curtis et al. 1988, Damian and Zowghi 2002, Herbsleb et al. 2005). It fails in the face of uncertainty (Kraut and Streeter 1995). It is even more difficult to convey precise meaning while defining requirements specifications if the target audience becomes more diverse across different roles and organisational cultures (Rönkkö 2007). Geographically dispersed collaborators are still forced to put more effort and rigour into processes and formal documents to try to improve team communication (DeLone et al. 2005). Gumm (2007), however, found this to be a benefit of GSD, since an improved focus on documentation is taken to be an improvement to the overall
development processes. There are therefore conflicting reports of the nature of documentation in GSD, and whether this is actually a challenge or benefit of the phenomenon.

2.5.3 Lack of informal and face-to-face communication

GSD literature has highlighted the lack of face-to-face communication as a major component of the approach to software development. Informal communication is a natural process of co-located development; it happens in hallways, over meals and before and after formal meetings (Herbsleb 2007). Informal communication helps a common language and organisational culture to form. People have visibility of what other people are working on, and the level of urgency of issues that may arise. Overall, informal face-to-face communication is a natural flow of communication that helps throughout the software development process.

Meanwhile, it has been reported that communication and control problems “are among the most troublesome and pervasive in software development” (Herbsleb et al. 1995). Both geographical and temporal distance reduce the opportunities for informal communication to take place, as both distances result in people not be located at the same place at the same time (Curtis et al. 1988, Grinter et al. 1999, Kiel 2003, Casey and Richardson 2004). Carmel (1999, p. 39) identified five centrifugal forces which “pull apart” globally-dispersed software development teams. Loss of communication richness is one of those forces, and is therefore a threat to the integrity of the team. The ability of teams to respond to changing requirements can be impacted upon by their lack of ability to communicate and coordinate due to the distances involved.

Indeed, “problems with projects can invariably be traced back to somebody not talking to somebody else about something important” (Beck 2000). In settings other than face-to-face interaction, there is a lack of immediacy and of communication richness (such as hand gestures) (Clark 1996). In GSD, breakdowns in informal communication could easily lead to misunderstandings in design conventions, and face-to-face contact is necessary to overcome misunderstandings of requirements (Curtis et al. 1988). For satellite sites where managerial decisions are not made, they may be “constantly surprised” by decisions, as they are not aware of decisions until they have been formally taken (Grinter et al. 1999). A lack of informal communication can mean that conflicting assumptions by developers at both sites are carried on until integration (Grinter et al. 1999).
The effect of difficulty to communicate may be more pronounced at certain phases of the software development. Kraut and Streeter (1995) found that informal face-to-face communication is more pronounced during the early stages of software development. During intensive problem-solving, design, or conceptual collaboration in GSD teams, many global software managers choose the channel of communication that allows for the richest communication (Carmel 1999, p. 48). This richness of communication is important, since words themselves are incomplete (Rönkkö 2007). Without proper knowledge of the context of words, it may be difficult to interpret the intended meaning of those words. Herbsleb, Klein et al. (1995) found that the design phase involves significant communication levels. Their study of a medium-sized (co-located) telecommunications software project found that in the first month of the project, 50 percent of time was spent in group work, but this gradually dropped to 10 percent.

2.5.4 Increased effort to initiate contact

Having team members separated by geographical distance places a barrier on communication by increasing the effort required to initiate contact (Herbsleb and Grinter 1999b). Distance increases the “cost” (or effort) of communicating due to increased difficulty, inconveniences and frustration (Grinter et al. 1999). Due to this effort, a software developer may carry on working without clarifying requirements specification, leading to further troubles later in the project (Rönkkö 2007). Without awareness of whether or not a remote colleague is at their desk, opportunistic interactions may be lost (Isaacs et al. 2002). In a co-located setting, a team member could simply glance over at their colleague’s desk to ascertain their level of availability.

This can lead to developers taking the risk of applying minor modifications to the system without trying to make contact with the person who might have more knowledge of that part of the system (Herbsleb and Grinter 1999a, Herbsleb and Grinter 1999b, Boland and Fitzgerald 2004). As a consequence, errors may be introduced in the system, ultimately increasing the cycle time. Furthermore, once the change is made to the system, the change may not be explicitly communicated to remote colleagues (Imsland et al. 2003), resulting in greater potential for errors to occur.

A related factor to this issue is that of team members not knowing who to contact (Herbsleb and Grinter 1999b, Battin et al. 2001, Braun et al. 2003, Herbsleb and Mockus 2003), and in a highly-dispersed project, it is likely that not all knowledge will be locally available (Grinter et al. 1999). Indeed, research has been conducted in the knowledge
management and knowledge transfer area specifically in relation to distributed software teams. Kotlarsky et al. (2008c) adopted a knowledge-based model to study coordination, taking the view that knowledge processes are required for cross-site coordination. They found that it is more useful to focus on knowing who knows what rather than knowing specific pieces of information. Oshri et al. (2008), building on (Wegner et al. 1991), define this type of knowledge as ‘transactive memory’. As transactive memory emerges from the social interactions of those involved, it has been classified as a form of ‘social capital’ (Rottman 2008). Kotlarsky et al. (2008c) advise on designing organisational structures that facilitate this knowledge flow because of the importance of this type of knowledge. In a separate study, Oshri et al. (2008) found that transactive memory could be shared through mechanisms that allowed for the systematic encoding of the subject and location of information to be exchanged. They also recommended selecting project managers for distributed projects based on their shared histories of collaboration so as to begin with a certain degree of transactive memory.

2.5.5 Cost of travel

Sometimes, meeting remote colleagues face-to-face is indispensable, especially in the early phases of a project (Herbsleb et al. 2005). In GSD, travel by team members to other sites allows for them to get to know their colleagues, for them to exchange ideas, and to break down cultural and linguistic barriers (Ebert and De Neve 2001, Kiel 2003). Having the chance to “drink a few beers” with remote colleagues can make huge improvements to their working relationships (Herbsleb et al. 2005).

However, this travel can be very expensive and time-consuming, with the travel time perhaps taking much longer than simply the flight time alone (Battin et al. 2001, Herbsleb et al. 2005). Also, depending on the extent of the geographical distance, there may not be direct flights between the two points of travel, increasing the journey time.

2.5.6 Dependency on information and communication technologies

In GSD, the dependency on information and communication technology (ICT) is high. Here, technology is used for communication and it therefore has an impact on the most critical processes in an organisation – whether and how people communicate to coordinate their processes (Sproull and Kiesler 1991, Karolak 1998). When a development team isn’t communicating well, it is nearly impossible to create and validate design solutions and effectively manage the team’s deliverables (Suzuki and Yamamoto 1999). Hence, a
convenient and well working technical infrastructure for information and communication, for example, effective tools and work environments, seems to be a necessity for successful GSD (Suzuki and Yamamoto 1999, Ebert and De Neve 2001). An adequately large inter-site network bandwidth is required to allow bandwidth-heavy uses such as video conferencing, and the need to e-mail large attachments (see Availability of technical infrastructure). One advantage of using ICT (including e-mail, instant messaging and fax) is that a written record can be kept (Damian and Zowghi 2002).

Common communication methods in GSD teams are e-mail, telephone, telephone conferencing, instant messaging and mailing lists, each technology offering a different degree of timeliness and content richness (Karolak 1998). The telephone may be the preferred method of (synchronous) communication for native speakers of the working language, but not for non-native speakers (Boland and Fitzgerald 2004). A telephone call may help end a long and ineffective e-mail chain (Paasivaara 2005). Conference calls may require some team members to work later or earlier than usual (Battin et al. 2001) – and practices of agreeing to working late or not can vary between countries (Brannen and Salk 2000). Due to socio-cultural distance, non-native speakers can find it difficult to keep up with the speed of the conversation, with topics having moved on by the time they have responded (Sarker and Sahay 2002). This can prove frustrating to those left behind (Kiel 2003).

2.5.7 Providing technical infrastructure

When developing software within a global context, problems can arise with global support for third-party tools being used (Grinter et al. 1999). Battin et al. (2001) found that different versions of tools were being offered in different countries by the third-party vendors, hindering the provision of a technical infrastructure. For example, the newest version of a tool was made available in the US, with older versions still being offered in other countries. Export regulations may prohibit diffusion of certain technology throughout the distributed team (Battin et al. 2001). The organisation may have to install its own private communication links, perhaps through expensive satellite links (Kobitzsch et al. 2001). An unstable electricity supply may cause data loss and destroy hardware. Security of the communication links may also be of concern. A common trust model such as PGP (Pretty Good Privacy) may need to be used to secure communication (Zimmermann 1995, Hayduk 2003). The quality of communication infrastructure in India is still a negative factor to doing business there (Kotlarsky and Oshri 2008).
2.6 Communication and Temporal Distance

2.6.1 Reliance on asynchronous communication technologies

When team members are not able to communicate in a synchronous manner (such as face-to-face, by telephone or by instant messaging), reliance on asynchronous communication technologies is increased. Since asynchronous communication relies on technologies such as e-mail and fax (Kiel 2003, Boland and Fitzgerald 2004), a written communication history is usually left (Carmel and Agarwal 2001, Damian and Zowghi 2002). This provides for increased traceability and accountability, in that it facilitates finding out who said what to whom, and when it was said (Ågerfalk 2004). These asynchronous communication technologies, such as e-mail, allow for input from diverse stakeholders, irrespective of geographical location (Damian and Zowghi 2002).

The reliance on asynchronous communication technologies can also act as a threat to communication and coordination. E-mail can “get lost” or “forgotten”, introducing an uncertainty of whether or not a reply will be forthcoming, and introducing the need to resend e-mails after a number of days (Imsland et al. 2003). Asynchronous communication can increase the amount of time it will take to resolve the issue at hand (Grinter et al. 1999, Boland and Fitzgerald 2004). Questions received overnight can be overwhelming for the developers beginning work in the morning (Boland and Fitzgerald 2004). Communicating through asynchronous methods such as e-mail does not provide adequate means for handling ambiguities, due to the lack of interactivity of the medium (Damian and Zowghi 2002). Furthermore, the conversion of ideas into e-mail form can also increase the risk of misunderstanding (Damian and Zowghi 2002), especially when the content of the communication is contentious or argumentative in nature (Kiel 2003). An ambiguous question posed in an e-mail can result in lengthy e-mail chains which are only broken by intervention by management (Paasivaara 2005). E-mail can also be misused as a weapon to publicly attack a peer, by copying senior management (Casey and Richardson 2004).

2.6.2 Delayed feedback

Temporal distance can result in team members not being at work when their help is needed (Kiel 2003). This introduces delays (Carmel and Agarwal 2001), rendering communication less effective (Karolak 1998). This increases the vulnerability costs of a development team, such as misunderstandings and rework needed to correct resultant mistakes (Espinosa and Carmel 2004). An issue can become exacerbated and can drag on over days. For example,
in a case study by Kiel (2003) of a single company with software development teams in
Canada and Germany, temporal distance presented a great challenge. Problems that should
have been simple to resolve often dragged on for days. “What might have been settled by a quick
conversation was often blown out of proportion because the information needed in order to resolved the
situation had to be communicated through e-mail, a message that might not be read for as much as 16
hours” (Kiel 2003). In another documented GSD project between a Norwegian company
and a Russian company, it took one to three days to receive a response to an e-mail
(Imsland et al. 2003).

A study by Herbsleb, Mockus et al. (2000) of software teams in Lucent Technologies
distributed across the UK, Germany and India found that temporal distance did introduce
delays to the work. They studied change request modifications on the versioning control
system used by the teams. Comparing dispersed and co-located teams, it was found that the
dispersed projects took 2.5 times longer to complete, because of delays in feedback due to
temporal distance. Perhaps the organisation was unable to achieve time zone effectiveness,
or that the cross-site coordination was not efficiently managed. In any case, the
organisation was not realising the benefits of time zone efficiency.

When awaiting feedback by e-mail, e-mail provides no indication of when an electronic
answer will be sent back: “you’re forced to rely on individual work styles and this may introduce
significant delay in resolving requirements issues” (Damian and Zowghi 2002). It should be noted
that while temporal separation does increase vulnerability costs, it also lowers coordination
cost when the project is running smoothly (c.f. 2.9.2 Time zone effectiveness).

2.6.3 Shifting work hours

A problem with trying to reduce the impact of temporal distance is that it’s inherently not
solvable while the team members are situated in different time zones (Carmel 2006).
Human beings stay awake during daylight hours due to deeply embedded biological and
social norms. Without modifying sleep patterns and/or work hours, actors separated by
large temporal distance are not able to communicate through synchronous communication
channels (Carmel 2006). Indeed, in Carmel’s study (2006) of the large Indian offshoring
service provider, Infosys, it was found that the company’s strongly enforced corporate
culture of flexible working hours (‘flexible’ for the organisation but not particularly for the
personal lives of its employees) and offices being open for twenty four hours of the day
were the main factors in facilitating its effective global operation. This approach may not
be feasible in the long-run, especially in countries with highly-competitive labour markets.
such as India where IT workers have a wide choice of jobs and may prefer those that offer normal working hours.

2.7 Communication and Socio-Cultural Distance

2.7.1 Asynchronous communication preferred by non-native speakers

Having to communicate in real-time over teleconferences can be overwhelming for non-native speakers, finding it difficult to keep up with the conversation (Sarker and Sahay 2002, Kiel 2003, Sahay 2003). Telephone conversations can involve complex technical or methodological debates – a necessary component of any software development activity (Kiel 2003). Developers have a general preference to deal through synchronous technologies for this reason (Boland and Fitzgerald 2004). However, asynchronous communication allows for non-native speakers to formulate their position and to check that they are making their point clear before sending the e-mail (Sarker and Sahay 2002). Thus, non-native speakers tend to rely more heavily on asynchronous communication. As previously identified, such communication can lack richness of context.

2.7.2 Language differences and misunderstandings

While English has become the international language for business matters, language competency is still a large stumbling block for communication within and between development teams (Imsland et al. 2003). In turn, misunderstandings can arise (Casey and Richardson 2004, Kotlarsky and Oshri 2008, Levina and Vaast 2008). Even if the whole team are native speakers of the working language, problems can arise from different dialects and local accents (Robey et al. 2000, Casey and Richardson 2004). Casey and Richardson (2004) reported of a GSD effort across Ireland and the US. American developers in their study found some strong Irish accents difficult to comprehend on the telephone. Furthermore, they reported that a difference in expressions between US and Irish dialects caused some misunderstandings. As reported in a study by Levina and Vaast (2008), the level of competency in English may differ between managers and developers within the same site. GSD teams may operate through English where none of the team members’ first language is English (Imsland et al. 2003). If a major section of the team speaks a particular language natively, a feeling of alienation can arise, with non-native speakers of the major language being at a disadvantage in expressing themselves (Kiel 2003, Sahay 2003). Software developers communicate through a common institutional language.
However, the understanding of the intended meaning of such language is affected by organisational and socio-cultural distance, since the understanding depends on culture, organisation, contexts, profession, and local politics (Rönkkö 2007).

2.7.3 Cultural differences and misunderstandings

Language is not the only source of difficulty arising from socio-cultural distance. There are cultural norms, values and styles of communication related to each national culture which can cause confusion when team members with differing cultural backgrounds try to communicate (Sahay and Walsham 1997). Kiel (2003) reported on a project that ultimately failed due to a mixture of social, cultural, linguistic and political factors. In the GSD context, geographical and temporal distance can multiply the effect of socio-cultural distance, making the process even more complex (Nicholson and Sahay 2001). Working norms can also greatly vary between organisations. In software development, this can mean that there is a fundamental difference in opinion about the nature of the software development process (Nicholson and Sahay 2001).

With the popularity of offshoring to Asia, especially to India, some differences between Western culture and Indian culture have been highlighted in the literature. Indian life is dominated by social classes, and therefore Indian workers may seem submissive and shy from a European perspective (Sahay and Walsham 1997). Sahay and Walsham found that even if an Indian team member is authoritarian in India, they can become submissive when located at the UK site. The submissive nature may be accentuated when dealing with aggressive and competitive colleagues at another location, with the Indians opting for accommodation rather than conflict. There seems to be a strong desire in Indians to please and to avoid confrontation. This may lead to a certain managerial issue that has been reported upon (Nicholson and Sahay 2001, Herbsleb et al. 2005). It has been reported that when a section of work is presented to Indian team members for completion, and asked if they can meet the deadline, they have a tendency of saying ‘yes’ even if they suspect that they cannot meet the deadline. This may be due to both the desire to accommodate others and also the sense of duty to the family and one’s superiors. In the study by Herbsleb, Paulish et al. (2005), the American team members understood answers comprising of ‘yes’ as agreement instead of the intended acknowledgement, which caused misunderstandings and acted as a barrier to communication.

Language is very much a part of a national culture, and is intertwined with social norms and values. Different communication styles may lead to misunderstandings. One example
of this was reported by Imsland, Sahay et al. (2003). The teams from Norway and Russia in their study were communicating through English. Misunderstandings began to arise as each team’s communication style differed due to their different national cultural backgrounds. The Norwegian developers came from a ‘low-context’ society, where intentions are generally explicitly stated without much reliance on context. On the other hand, the Russian developers communicated in a ‘high-context’ style, where intentions can be implied rather than explicitly stated. Nonverbal communication is particularly important for the high-context cultures outside of Anglo and Northern European cultures (Carmel 1999, p. 48).

2.7.4 Managing frames of reference

Frames of reference held by team members serve as guidelines to organise and shape their interpretations of events within the organisation and to give these meaning (Orlikowski and Gash 1994). Culturally- and geographically-distributed teams must achieve mutual understanding in order to work together effectively. For the intended meaning of any set of words to be understood, it is required that the receiver has both knowledge of the context in which those words were uttered or written, and has knowledge of previous conversations, the producer’s normal behaviour, and of common communication practices expected by the producer (Rönkkö 2007). Establishing mutually-held knowledge is important since it increases the likelihood that communication will be successful (Cramton 2001). The process of communication allows for team members to form a more cohesive understanding of the work involved (Armour 2002). The most effective teams will develop a common ‘language suite’ comprising of in-jokes, and a code of conduct that governs their rules for behaviour (Armour 2002).

This issue deals with several distances. Socio-cultural distance may be one source of a fundamental mismatch of frames of reference. As also identified, geographical distance can be a barrier to developing shared frames of reference. As such, co-location is most effective to allow these to form (Armour 2002). GSD therefore is an obstacle to this formation. At the very least, each actor must have an understanding of the others’ frames of reference (Ågerfalk et al. 2005).
2.8 Coordination and Geographical Distance

2.8.1 Access to large skilled labour pool

GSD provides an unprecedented possibility to access large skilled labour pools by coordinating across distance, reaching into new labour markets (Grinter et al. 1999, Herbsleb et al. 2000, Carmel and Agarwal 2001, Herbsleb and Moitra 2001, Damian et al. 2003, Kotlarsky and Oshri 2008). Highly-skilled software developers can be difficult to find locally in the higher-cost economy (Herbsleb et al. 2000). By implementing GSD, organisations can include the contributions of thousands of skilled workers in new offshore locations where available skilled staff are plentiful, allowing for the quick formation of virtual global corporations (Suzuki and Yamamoto 1999, Battin et al. 2001, Herbsleb and Moitra 2001, Carmel and Tija 2005). This relies on skills being available in due time (Ebert and De Neve 2001). GSD allows organisations to cost-effectively capitalise on the global resource pool, use scarce resources, wherever they are located, and within due time (Ebert and De Neve 2001, Herbsleb and Moitra 2001). The implication of this benefit of GSD is that people are hired in different labour markets, and must then coordinate with other teams across the world. The coordination must take place across geographical, but also temporal and socio-cultural distances. This benefit can also be related to socio-cultural distance if the different cultures can contribute positively to the overall project. This is further discussed in relation to innovation and shared best practice in a later section.

2.8.2 Access to cost-effective labour pool

Cost is the main and original driving factor of GSD – more emphasis is placed on cost savings than any other potential benefit of GSD (Kobitzsch et al. 2001, Garrad 2003a, King 2003, Levina and Ross 2003, Global Insight (USA) Inc. 2004, Carmel 2006). In their review of literature of IT outsourcing, Lacity et al. (2009) found that cost reduction was by far the most cited motivation for outsourcing. Organisations can access skilled workforces where average comparable salaries are much lower than in the home country, allowing reduced development costs (Karolak 1998, Carmel and Agarwal 2001, Herbsleb and Moitra 2001, Damian et al. 2003, Greenhouse 2003, King 2003, St.Amant 2005, Meyer 2006). However, expected cost savings may not be fully realised because of extra hidden costs (Kharif 2003).

The difference in wages across geographical distance can be significant, with a US worker’s salary being multiple times greater than that of a worker in Asia. For example, Levina and
Vaast (2008) found in 2005 that pay levels for technical leads in the US stood at $80,000, then $9,000 in India, and $18,000 in Russia. Such pay levels would indicate an up-front saving in eight times the cost in India compared to the US. Other similar figures have been reported elsewhere (although it is difficult to offer a direct comparison as different studies have reported on various I.T. roles). For example, the annual base salary for a software development engineer was US$10,300 in India and US$13,400 in China in 2005 (Mercer 2005). According to the US Bureau of Labor Statistics, the median annual salary of a US computer applications software engineer was nearly US$75,000 in 2005 (Bureau of Labor Statistics - U.S. Department of Labor 2005). Previously in 2003 it was reported that an annual salary for an Indian systems programmer stood at US$6,000 (Rottman and Lacity 2004). The annual base pay may be supplemented with additional bonuses. A US systems programmer earned an average of US$63,331 in 2003 (Rottman and Lacity 2004). Kotlarsky and Oshri (2008) note that India’s success as a low-cost market for software development still holds true.

The chance to leverage cheaper employment markets has opened up a wide range of possibilities. Sourcing IT services globally has become a fundamental transition, a “megatrend” (Greenhouse 2003). Organisations are presented with global opportunities to tap into available pools of skilled labour “to exploit the power of communication technologies to create new sources of competitive advantage” (Venkatraman 2004). The opportunity provided by cost savings in software development has facilitated a new generation of software development on a global scale, and presents practitioners with additional benefits such as access to affordable multi-skilled labour force.

2.8.3 Reduced trust

Trust plays an important role in the development of personal networks and personal relationships (Boutellier et al. 1998). Mutual trust is important for people to be able to fully cooperate and work with their colleagues. As such, a lack of trust could result in the breakdown of coordination efforts between remote teams. Pyysäinen (2003) defines trust as “a relationship between parties”, rather than trust being the property of an individual. Trust prevails if each of the interacting parties acknowledges the right of the other parties to assess the competence and the intentions of their acts (van der Smagt 2000).

---

1 Given the dynamics of the software development market, salaries may have fluctuated considerably since these published figures. Since the nature of data being reported may vary between countries, any comparison here is only indicative.
Imsland (2003) noted that trust is a complex phenomenon when looking at the role of trust in global software outsourcing. He identified seven attributes of trust to try to explain the phenomenon: predictability, competence, structure, calculation, goodwill, knowledge and betrayal. He noted that the particularities of trust depend on the context, such as intra-organisational and inter-organisational relationships, and whether at a micro level between individuals or at a macro level between organisations.

Trust takes time to establish (Carmel 1999, p. 53). It requires close interaction with the other party in order to develop (Imsland 2003). Long-time interaction increases the knowledge about the partner, making their behaviour more predictable. For example, a software developer can learn of the competencies of their co-located colleagues, allowing them to place trust in their future work. The distances in GSD hinder the creation of trust, since normal communication like face-to-face feedback and common experience are sources of trust (Pyysiäinen 2003). Co-located team members can build trust through formal and informal face-to-face meetings (Carmel 1999, p. 53). Familiarity and confidence are stages of relationships that must take place before trust is formed. Achieving and maintaining trust in global teams is therefore more difficult than in co-located teams (McDonough et al. 2001). Newly-established dispersed teams take longer to get through the preliminary stages of team development (Carmel 1999). Dispersed teams hardly ever meet their remote colleagues, creating a barrier to the establishment of trust. At a distance, it is difficult to empathise with those at the other site (Kiel 2003). Socio-cultural distance may make others’ actions less predictable, which may also hinder the development of trust. Trust can also be corroded, for example when defects are introduced due to a developer not making the effort to contact a remote colleague before making changes to the system (Boland and Fitzgerald 2004). When there is a lack of trust, there is a lack of willingness to communicate (Herbsleb and Grinter 1999b) and the amount of information disclosed to remote colleagues may be limited (St.Amant 2005).

2.8.4 Lack of awareness/team spirit

A team is an organisational unit in itself, and it brings with it organisational benefits such as the synergy of ideas and innovation, improved problem solving, and the sharing of expertise and experience (Carmel 1999, p. 50). As such, the structure of being a team is important to maximise the effectiveness of the people working to coordinate to solve a software problem. However, the feeling of “teamness” with remote colleagues can be affected because of physical separation and lack of informal contact (Carmel 1999, Battin et
al. 2001, Herbsleb and Mockus 2003, Kiel 2003). Presumably, distance affects the stages by which individuals become coherent groups or teams (Carmel 1999, p. 54). The lack of teamness, and the a sense of separation, can be exacerbated due to cultural distance between both sites, e.g. cultural distance between Norwegian and Russian developers (Imsland et al. 2003). This sense of separation can lead to an imbalance of resources between ‘onshore’ and ‘offshore’ teams (Levina and Vaast 2008).

Due to physical separation and lack of face-to-face contact, team members may not be aware of the details of their remote colleagues’ work activities, and engineers may begin to handle their work products inefficiently (Ebert and De Neve 2001, Armour 2002). If team members are unaware of colleagues’ work, misunderstandings can continue unnoticed and code conflicts can later arise. This results in unexpected delays when code is delivered from one site to another (Herbsleb et al. 2005).

It can be difficult to determine if a remote colleague is available to be contacted at a particular time (Herbsleb and Grinter 1999b), without a visual cue such as open doors or having knowledge of local schedules and holidays (Grinter et al. 1999). Team members may feel that remote colleagues do not help them as much because they are ‘out of sight’ (Grinter et al. 1999, Herbsleb and Mockus 2003). It is perceived that remote colleagues are much less likely to provide information quickly, to follow up on requests, and, generally, to consider the concerns of the other side (Grinter et al. 1999). A lack of trust across sites may be accompanied with a lack of mutual respect between remote colleagues (Kiel 2003).

All of these problems culminate in a lack of sense of “teamness” across sites. Cross-site travel has been advocated to help generate a sense of teamness (Herbsleb and Grinter 1999b). However, travel restrictions may not allow for this.

2.8.5 Cross-site modularisation of work

According to Conway’s Law, the structure of the system mirrors the structure of the organisation that designed it (Herbsleb and Grinter 1999b). In earlier work we have seen the importance of a separation of concerns when decomposing work into modules (Parnas 1972), and it appears that these principles could be extremely relevant for managing coordination complexity.

The nature of GSD leads teams to splitting their work across feature content into well-defined independent modules (Ebert and De Neve 2001, Sahay 2003, Bass and Paulish 2004). This allows decisions to be made about each component in isolation (Herbsleb and
Grinter 1999b). Partitioning work tasks horizontally resulting in each site having responsibility for the whole lifecycle of particular functions/modules decreases interdependencies, and hence coordination costs (Battin et al. 2001). Source code branching enables software development teams to work on source code in parallel, and merging the sections once they have been developed (Herbsleb et al. 2005).

At the same time, teams need to be wary of reduced communication between sites, leading to problems at the integration phase (Grinter 1998). “Big bang” integration can occur when many outstanding problems remain unresolved (Battin et al. 2001). Remote teams may become extremely isolated in strongly modularised projects, and this can result in huge overhead upon project completion or iteration (Ebert and De Neve 2001, O’Leary and Cummings 2007). Integration also needs “fast-paced” face-to-face communication (Herbsleb et al. 2005), which can only be provided through temporary co-location (Ebert and De Neve 2001).

2.8.6 Lack of mechanisms for creating shared understanding

Without effective mechanisms for sharing information and facilitating common understanding across distances, managers cannot exploit the benefits of GSD (Herbsleb and Moitra 2001). A failure to establish such mechanisms may result in knowledge and experience asymmetries between different sites (Vlaar et al. 2008). Inadequate dispersal of important information about a project, such as the overall architectural vision, can leave teams with a skewed perception of which tasks are on the critical path (Battin et al. 2001). With a lack of understanding of the wider system, opportunities to re-use software code may be overlooked (Herbsleb and Moitra 2001). Software architects have a responsibility for creating a “shared project context” in allowing distributed team members to understand the nature of the work being distributed (Raghvinder and Johannes 2008). This issue comes back to the fact that richness of communication is limited in a distributed context: if rich communication is required to create a shared understanding, then communicating through communication technologies may hinder the development of shared understanding. On the positive side, if a project does manage to effectively create shared practices across sites, then collaboration between sites is made easier (Levina and Vaast 2008).

In certain circumstances, it has been possible to create such a shared understanding. In particular, the Jazz distributed development project at IBM (Wolf et al. 2008) has been able to create a sense of shared understanding and shared vision resulting in a successful outcome. However, the authors speculate that not only were they able to create shared

32
collaborative mechanisms in a distributed environment, but the fact that the project involved creating a product for distributed integrated programming environment helped their work, and such strong mechanisms may not always be available for more general software development projects. Still, the study highlights the possibility of improvement through the development of mechanisms of shared understanding compared to other less successful reported studies.

2.8.7 Improved process definition

Independent of a project’s process maturity, the definition and structuring of processes is a challenge (Gumm 2007). While distributed project settings can challenge process maturity, they also seem to support its development. Gumm (2007) found that process definitions are compiled more carefully in distributed settings. It was noted that if team members were co-located, much of the processes would probably not be formalised.

These standardised practices, including manuals, tools and implicit and undocumented systems, serve as points of reference to coordinate work across time and space (Robey et al. 2000). Even using the same programming language can help develop a common understanding (Armour 2002). A study by Herbsleb, Paulish et al. (2005) found that failing to invest in common tools due to financial restrictions was a false economy. Problems have also been reported for teams where management did not apply common processes across sites (Battin et al. 2001). As a counter point, allowing local variation in work practices may leverage local expertise and reduce project overhead (Akmanligila and Palvia 2004).

2.8.8 Increased effort in outsourcing arrangements

With respect to coordination complexity and outsourcing specifically, Barthélemy (2001) identified four areas of added costs specific to outsourcing to a third party vendor:

1. Vendor searching and contracting (c.f. Overby 2003a)

2. Transitioning to the vendor

3. Managing the effort

4. Transitioning after outsourcing

These transaction costs were similarly classified as ‘contact, contract, and control’ by Carmel and Nicholson (2005). In the survey reported by Barthélemy, few companies had
thought about all the costs of transitioning to a new vendor, and had assumed that the cost of the efforts would be negligible. Organisations should assess the risk of knowledge loss before partaking in an outsourcing relationship (Deloitte 2005).

2.8.9 Perceived threat from low-cost alternatives

Employees in the higher-cost economies can feel that their jobs are under threat from their colleagues in lower-cost economies, creating a “we versus they” mentality (Herbsleb and Grinter 1999b, p. 93, Baker et al. 2004, Casey and Richardson 2004, Minevich and Richter 2005). They may see a threat to their future employment and promotion prospects – the ‘my job went to India and all I got was this lousy T-shirt’ syndrome.

The GSD phenomenon has been a hotly debated topic in the US, where the terms “outsourcing”, and especially “offshoring”, have been widely covered in the media, much with negative coverage. Such headlines as “Software: Will Outsourcing Hurt America’s Supremacy?” have appeared in the American press (Baker et al. 2004). In 2003, Greenhouse (2003) stated that a company announcing that it will move jobs overseas in order to cut costs can expect a backlash among politicians and even its own employees, including “anger” from government officials. Greenhouse noted that the trend to offshore jobs was not limited to IT jobs. Some had seen this as “stealing jobs” from US workers (Rottman and Lacity 2004). Forrester research has estimated that 450,000 computer industry jobs in the US could be transferred offshore between 2003 and 2015 (Greenhouse 2003).

When a company offshores some work, a significant wage differential appears between the original and new development sit. US-based employees may feel threatened by their remote colleagues, who earn much less than they do, and who, in their eyes, may be a threat to their livelihoods (Herbsleb and Grinter 1999a, Casey and Richardson 2004). As a result, they may not want to cooperate with their remote colleagues.

The publications identified in relation to this challenge show a peak of interest in the period of 2003-2005. It is possible that the US public has softened in its view of offshoring of IT-related work if it has become more accustomed to the phenomenon. Outsourced work may be more maturely viewed as a method of completing work where in-house staff do not have sufficient time to complete the work, and therefore be seen as an opportunity for the company (Richardson et al. 2008), rather than as a threat. As such, this identified challenge may or may not still have an effect on US-based companies making strategic
decisions in relation to GSD. This is also a socio-cultural issue, as a lack of familiarity with other cultures may exacerbate the “us versus them” perception.

2.9 Coordination and Temporal Distance

2.9.1 Reduced hours of collaboration

A disadvantage of being separated by temporal distance is that the number of overlapping hours during a workday is reduced between sites (Battin et al. 2001, Kiel 2003, Casey and Richardson 2004, DeLone et al. 2005). For example, a team located across the both the eastern U.S. and in Ireland can have a total of three overlapping hours during a workday (Casey and Richardson 2004).

Even a one-hour time zone difference can mean a large reduction in overlapping hours. For example, with team members in Germany working from 8am-4pm with a 12pm lunch, coordinating with UK team members working from 9am-5pm with a 1pm lunch, there are only four overlapping hours in a work day (Herbsleb and Grinter 1999b).

This issue can lead to delays in feedback, and to an increase in coordination complexity. In contradiction to this, temporal distance may be leveraged to a team’s advantage as discussed in the following section.

2.9.2 Time zone effectiveness

By communicating in an asynchronous manner, teams can cover more working hours during the working day, thereby increasing temporal effectiveness (Ebert and De Neve 2001). Although the face-to-face setting is the basic prototype for communication (Fillmore 1981, Carmel 1999, Casey and Richardson 2004), and is considered to be a “richer” form of communication (Carmel 1999, Herbsleb et al. 2000), asynchronous communication across temporal distance can be leveraged to the distributed team’s advantage. The approach can aid organisations which are under severe pressure to improve time-to-market (Herbsleb and Moitra 2001).

The extreme of time zone effectiveness has been seen as an advantage for GSD, with the promise of a “follow-the-sun” model of development whereby teams coordinate to cover 24 hours of the working day (Carmel 1999, Herbsleb and Grinter 1999b, Sarker and Sahay
One team can hand-off work at the end of their day to another team, and that next team can work on fixing problems whilst the first team sleeps (Sarker and Sahay 2002).

Follow-the-sun depends on closely coordinating tasks across time zones, increasing coordination complexity (DeLone et al. 2005), and involves other challenges such as delayed feedback from remote colleagues. Indeed, follow-the-sun development may be too difficult to implement – daily hand-offs of source code is too much to digest in one day’s work (Carmel 1999, p. 27).

Literature on GSD does not seem to have settled on agreement on whether or not time zone effectiveness is an achievable benefit. For example, Carmel and Agarwal (2001) have suggested reducing temporal distance to alleviate the challenges of distance in GSD. This would cancel out benefits of time zone effectiveness.

2.9.3 Time zone efficiency

While time zone effectiveness deals with maximising the number of hours during which software is being developed, time zone efficiency refers to the potential increase in efficiency given that temporal distance reduces the level of on-going synchronous communication.

Temporal distance can be seen as beneficial in terms of coordination, in that coordination costs are reduced when team members are not working at the same time (Espinosa and Carmel 2004). The producer of a unit of work can complete the work during the off-hours of the person who requested that work. In essence, coordination costs are reduced since no direct coordination takes place when two people are not working at the same time. In a separate study, Gumm (2007) reported that both temporal and geographical distance made life easier for the respondents. Specifically, the study found that the person asking the question (communicating across distances) would have thought more about the problem before making the effort of communicating across distances. This reduced the amount of unnecessary communication. Respondents also preferred that they could retreat to their own desks without being “bothered” by colleagues who were located remotely. Temporal distance gave them better opportunities for thinking intensively about a problem without being contacted by others.
2.9.4 Synchronous team meetings difficult to arrange

Team members might have to work flexible hours in order to coordinate with their remote colleagues through real-time teleconferences, increasing the cost and effort of coordinating regularly (Battin et al. 2001, DeLone et al. 2005). Cross-continental meetings are always difficult, as someone in one of the locations must always compromise on their work schedule (Damian and Zowghi 2002).

2.9.5 Availability of technical infrastructure

Lacking technical infrastructures, and possible incapability, greatly affect performance of GSD teams (Kobitzsch et al. 2001). Technology is used for synchronous communication across geographical distance and for asynchronous communication across temporal distance. It therefore has an impact on the most critical processes in an organisation – whether and how people communicate to coordinate their processes (Sproull and Kiesler 1991, Karolak 1998). Coordinating transparently across sites worldwide requires advanced communication infrastructure (Kobitzsch et al. 2001). According to Herbsleb and Grinter (1999a), distributed organisations desperately need tools that make it easier to find organisational information, to maintain awareness about the availability of people, and to have more effective cross-site meetings, especially spontaneous ad hoc sessions. Software development teams may find their development tools to be inadequate for their development activities, especially in a distributed environment (The Economist 2004). For example, most version control tools for software code do not allow 24/7 access without disturbing engineers due to back-ups and synchronisations (Ebert and De Neve 2001). More generally, the shared project database – the developed product and its documentation, process and measurement data, re-use repositories, and so forth – must be available to all team members (Kobitzsch et al. 2001). A slow network connection may result in delays for the completion of tasks (Herbsleb et al. 2005).

2.9.6 Coordination complexity

Software development in itself is a complex task with substantial non-routine work. The coordination process itself can be costly (Espinosa and Carmel 2004). The managerial effort required can be underestimated by the organisation (DiamondCluster International Inc. 2005). Substantial non-routine software development requires a fair amount of communication to coordinate (Espinosa and Carmel 2004). GSD can be ineffective since “project costs increase in proportion to the time it takes for people to understand each
other” (Cockburn 2002, p. 81). It’s important to manage team resources closely (Imsland et al. 2003, Espinosa and Carmel 2004). In a GSD team, it is necessary to synchronise roadmaps and deliverables across sites, and this requirement adds complexity to the development process (Ebert and De Neve 2001).

Companies starting to implement GSD strategies soon realise that time zones introduce additional coordination costs that negate some of the labour savings expected (Carmel 2006). Degradation of the quality or level of communication will affect efforts to coordinate dependencies. Coordination is costly (Malone and Crowston 1994), especially for distributed software development (Kraut and Streeter 1995, Espinosa et al. 2003b, DeLone et al. 2005). In this case, ‘cost’ refers to the effort involved in coordinating activities, a meaning adopted by DeLone et al. (2005) in their study of distances in GSD. The added complexity diverts attention from productive activities (Espinosa and Carmel 2004). For example, temporal distance becomes a substantial collaboration barrier and often results in time delays (DeLone et al. 2005). Cultural distance further impacts productivity (Overby 2003b).

2.10 Coordination and Socio-Cultural Distance

2.10.1 Innovation and shared best practices

Organisations can take advantage of increased innovation and shared best practice that arises from the collaboration of team members who come from different national and organisational backgrounds (Ebert and De Neve 2001, Carmel and Tija 2005). Developers from different cultural backgrounds work together to continuously improve a product and to improve processes, allowing for cross-fertilisation of best practices (Ebert and De Neve 2001).

In large complex organisations, decentralised independent individuals interact in self-organising ways to create innovative and emergent results (Highsmith and Cockburn 2001). Such organisations base their success on innovation, and that innovation comes from talented individuals – from the most brilliant, intelligent and creative engineers in the organisation. Companies that expand into other countries in order to tap into talent have been termed “knowledge seekers” (Chung and Alcacer 2003). Such organisations tend to act somewhat differently to organisations that offshore purely for cost reasons (Carmel and Tija 2005).
During the mid- to late-20th century, companies would have tapped into talent in other high-wage industrial countries. In the 1990s, however, software companies began to turn to countries such as Israel and India for employing talented engineers. The global business environment continues to change at a dramatically increased speed, and for companies to survive they need to innovate relentlessly (Highsmith and Cockburn 2001). Some companies outsource their research and development (R&D) activities to third-party vendors. By 2003, 77 global software product firms had established direct R&D subsidiaries in India (Carmel and Tija 2005, p. 11). Acquisition may be one viable method to tap into existing talent. Acquisition allows an organisation to access new markets, new products, new engineers and add creativity to existing teams (Ebert et al. 2001).

Organisations that use offshore development teams may benefit from the best practices of those teams. When using offshore third-party software vendors, there is a need for the organisation to coordinate work processes, particularly with suppliers who are committed to the Software Engineering Institute’s Capability Maturity Model (CMM). Indian companies, for example, are commonly certified at CMM level 4 or 5 (where 5 is the highest attainable level) (Rottman and Lacity 2004) which is often higher than that of the client (Meyer 2006). A gap in practices between customers and suppliers (suggested by the differing CMM levels) can cause problems due to incompatible processes. However, in such a situation, the customer may focus on raising the CMM level of their processes to reflect the supplier’s, hence resulting in the beneficial improvement of processes of the onshore teams. However, it should be noted that a high CMM level does not directly imply improved innovativeness. Moitra (2001) argues that it is unfortunate that Indian software firms are using their certifications as a marketing instrument. The drive for attaining a particular certification or CMM level has led to a predominantly compliance-based approach instead of really driving business excellence and innovation through quality and processes. Therefore, companies that wish to outsource their product development to third-party offshore vendors should be aware that this will not necessarily lead to innovative solutions.

2.10.2 Language and cultural training

Language and cultural training is a cost for GSD organisations. An investment in language training and cultural awareness may be required if team members come from different backgrounds (Imsland et al. 2003, Espinosa and Pickering 2006). Krishna et al. (2004) suggest compromising to establish an agreed working culture, in effect reducing cultural
distance. A “bridgehead” (Carmel and Agarwal 2001) or “liaison” (Battin et al. 2001) may be helpful, i.e. a person from one site working in another site and acting as a mediator between sites.

2.10.3 Lack of domain knowledge

Project development tasks can require specific domain knowledge that developers coming from different backgrounds do not have (Battin et al. 2001, Imsland et al. 2003). Domain knowledge can come from having worked on previous projects (Perry et al. 1998, Damian and Dustdar 2005). Organisations can have incompatible views on a domain, based on their own particular experience and expertise (Curtis et al. 1988). The underlying issue is that it can be difficult to convey implicit knowledge even with developers’ access to formal education and a full technology toolset (Leonardi and Bailey 2008). Imsland found that the Russian developers working on software that dealt with Norwegian tax rules could not fully understand the concepts involved (Imsland 2003, Imsland et al. 2003). In that study, the importance of the intrinsic knowledge possessed by the Norwegian developers had not been recognised before involving the Russian site, resulting in near-failure of the project. Imsland found that the coordination between the two sites was severely affected due to this lack of shared understanding of the domain.

2.10.4 Doubtful of others’ capabilities

In an issue closely related to lack of trust, developers may be doubtful of the knowledge of team members from other sites, their capabilities and skills (Grinter et al. 1999, Battin et al. 2001). It has, for example, been reported that American engineers can have concerns about the competency of international engineering teams (Battin et al. 2001). Robey et al. (2000) found that this doubt of others’ competencies can come from socio-cultural distance in that people form stereotypical impressions of others. This impression may be overcome by promoting familiarity and trust between teams (Robey et al. 2000). As with the potential challenge of perceived threat from low-cost alternatives, this challenge may have subsided given the maturity of GSD in the past few years.

On an organisational level, offshore outsourcing vendors providing offshore outsourcing services can also find it difficult to convince American and European companies of their capabilities. This results in their striving to achieve very high levels in process improvement models such as CMMI (Capability Maturity Model Integration) in order to prove their capabilities to potential clients (Meyer 2006).
2.11 Control and Geographical Distance

2.11.1 Allocation of roles and team structure

With the help of modern communication technologies, GSD can enable an organisation to build project teams using skilled people from different sites all over the world. Project management can be made effective by re-allocating people’s tasks for short periods of time, independent of how the project is globally distributed (Ebert and De Neve 2001). This flexibility helps alleviate issues with isolated expertise.

By employing new lower-cost labour, the organisation can re-assign high-skilled workers to other, often more strategic activities (DiamondCluster International Inc. 2005). This approach also avoids potential employee turmoil and backlash associated with workforce reductions. Carmel and Tija (2005, p. 11) argue that the abundant supply of skilled labour allows an organisation to assign many engineers to one problem, to focus on several projects at one time, and to ramp-up operations within days or weeks rather than months or years. Where restrictive headcount limits may be in place at high-cost sites, organisations may outsource mundane work to lower-cost locations, thus allowing the home engineers to focus on core tasks (DiamondCluster International Inc. 2005).

2.11.2 Lack of concurrent software engineering principles

Concurrent engineering (CE) principles have been better adopted in hardware development than in software development. Concurrent software engineering (CSE) principles is the marrying of CE principles to software engineering. CSE is defined as a management technique to reduce the time-to-market in product development through simultaneous performance of activities and processing of information (Blackburn et al. 1996). Such principles would allow for work to be better modularised allowing for parallel development of sections of the project. When GSD sites are communicating through mostly asynchronous communication methods because of temporal distance between them, synchronisation is important when the teams hand off processes between sites. It requires commonly defined milestones and clear entry and exit criteria. Effectively implementing concurrent engineering principles in GSD often becomes difficult because of volatile requirements, unstable specifications, the unavailability of good tools that support collaboration across time and space, and the lack of informal contact (Herbsleb and Moitra 2001).
2.11.3 Increased autonomy

A study by Gumm (2007) found that organisational and geographical distribution of software development units implies a certain degree of autonomy for each unit due to their physical isolation from each other. The study reported that this autonomy allowed for the necessity to maintain the different working cultures of each team. This was viewed as necessary in order to maintain the quality of the work of a single team even if this in turn required careful synchronisation of the individual processes.

2.11.4 Organisational risks

Depending on the locations involved, offshoring can increase the level of risk involved for the organisation. Carmel and Tija (2005, p. 45) state that the biggest risk when offshoring is country risk, which encompasses both political and financial risk. GSD may place parts of the organisation in developing or emerging countries, which have historically been more volatile, less stable, less predictable, and less transparent. Political risks include war, terrorism, rioting, uprising, military coups, confiscation, expropriation, and currency crises. Various reports have attempted to classify target countries for GSD, according to political stability, amongst other factors (Betts 2003, Carmel and Tija 2005, p. 69, Minevich and Richter 2005). Also of note is sovereign risk, where government regulations can change. Taxes that countries once used to attract business may be subject to increases.

Carmel and Tija identify eight additional risks that may affect the predictable regulation of offshoring efforts:

1. Intellectual Property (IP) risk. Enforcement in target developing nations of IP breaches are mostly rare. Software code and ideas (trade secrets) may be stolen, with little possibility of recourse.

2. Loss of proprietary knowledge risk. Critical know-how may be leaked to competitors, and this can more easily happen away from the company’s home country. This risk may be realised in the long-term, and therefore may not be obvious to companies with short-term offshoring plans.

3. Data security risk. The European Union and the United States government have imposed increasingly strict rules to protect private data. Companies must take special care when the sensitive data is available to their employees globally.
4. **Corruption risk.** This can affect offshore subsidiaries, but is less of a problem when practicing offshore outsourcing, since the vendor will absorb resultant costs.

5. **System security risk.** Offshore employees may be given substantial access to the corporate network, exposing the company to potential data leakages.

6. **Contractual risk.** If a dispute arises, especially in an offshore outsourcing context, a resolution may take longer to be achieved, and may favour the local company above the foreign (outsourcing) company.

7. **Infrastructure risk.** The communications infrastructure in some countries may be quite unreliable. Multiple communication links between sites may be needed.

8. **Societal and regulatory changes in company’s home country.** A political backlash may be experienced due to offshoring much work out of the home country. The fear of a poor public image then begins to drive decision-making.

These risks are in addition to the operative challenges that occur in GSD, such as added coordination complexity and lack of informal contact. These risks may erode the competitiveness (according to resource-based theory) offered by the large skilled labour pools available in developing and emerging nations.

### 2.12 Control and Temporal Distance

#### 2.12.1 Management of project artefacts

When a GSD project involves members from different organisations or organisational units, enforcing process and artefact standards can be particularly important in maintaining consistency and interoperability between project artefacts (Evaristo et al. 2004). To maintain consistency among project artefacts, a configuration management tool with centralised storage is often used. Artefacts include source code and project documentation. Even when working from the same central repository, it may be unclear what problems are addressed by a new version of an artefact and what status it is in (such as whether it is still being tested) (Braun et al. 2003). While this issue is not unique to GSD, it is accentuated by reduced levels of formal and informal communication levels that would help such uncertainties to be answered quickly. As already discussed, temporal distance can be an obstacle as there may be a delayed response from remote colleagues to clarify such issues.
2.13 Control and Socio-Cultural Distance

2.13.1 Adapting to local formalised norm structures

When working in a global setting, companies must learn about local formalised norm structures (applicable laws, import and export laws, traditions, regulations, etc). Labour may not be able to freely move between sites due to immigration laws – applications for visas and work-permits may need to be sent some time before a trip between sites (Battin et al. 2001). Local governmental regulations may require some local development activities in order to allow organisations participate in those markets (Herbsleb et al. 2000). Export laws may prevent teams outside of the US using tools that are in use by the US team (Battin et al. 2001). Also, different sites may prefer different development methods (Evaristo et al. 2004). (Akmanligila and Palvia 2004)

2.13.2 Different perceptions of authority/hierarchy

The nature of authority in a team environment can vary between cultures (Sahay and Walsham 1997, Krishna et al. 2004, Levina and Vaast 2008). It has been found, for example, that Irish developers require their superiors to earn their respect, while US developers are more submissive to authority (Casey and Richardson 2004). In another case, a Dutch company decided not proceed with developing software in India due to the estimation that male programmers in India would struggle to work under the supervision of a female executive (Kotlarsky and Oshri 2008). Problems may arise in controlling a GSD project if teams in different locations expect to be managed differently and if these different expectations are not identified early on.

2.14 A framework of the challenges and opportunities in GSD

Following the literature that has been reviewed in this chapter, the framework of the challenges and opportunities of GSD is presented in Table 2-2. This is an extension of the work that was co-published in the workshop paper by Ågerfalk et al. (2005).
<table>
<thead>
<tr>
<th>Process</th>
<th>Geographical Distance</th>
<th>Temporal Distance</th>
<th>Socio-Cultural Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Communication</strong></td>
<td>✓ Proximity to market/customer ✓ Improved focus on documentation × Lack of informal and face-to-face communication × Increased effort to initiate contact × Cost of travel × Dependency on ICT × Providing technical infrastructure</td>
<td>✓ Reliance on asynchronous technologies – record kept × Delayed feedback × Shifting work hours</td>
<td>× Asynchronous communication preferred by non-native speakers × Language differences and misunderstandings × Cultural differences and misunderstandings × Managing frames of reference</td>
</tr>
<tr>
<td><strong>Coordination</strong></td>
<td>✓ Access to large skilled labour pool ✓ Access to cost-effective labour pool ✓ Cross-site modularisation of work ✓ Improved process definition × Reduced trust × Lack of awareness/team spirit × Lack of mechanisms for creating shared understanding × Increased effort in outsourcing arrangements × Perceived threat from low-cost alternatives</td>
<td>✓ Time zone effectiveness ✓ Time zone efficiency × Reduced hours of collaboration × Synchronised team meetings difficult to arrange × Availability of technical infrastructure × Coordination complexity</td>
<td>✓ Innovation and shared best practices × Language and cultural training × Lack of domain knowledge × Doubtful of others’ capabilities</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>✓ Allocation of roles and team structure ✓ Increased autonomy × Lack of concurrent engineering principles × Cannot manage by walking around × Organisational risks</td>
<td>× Management of project artefacts</td>
<td>× Adapting to local formalised norm structures × Different perceptions of authority/hierarchy</td>
</tr>
</tbody>
</table>

The basis of the framework are the processes of software development (communication, coordination and control), and how they are affected by geographical, temporal and socio-cultural distances. The “✓” symbol is used in Table 2-2 to denote an item that has been identified as a benefit of GSD, whereas items with an “×” are potential challenges. Not all issues can neatly lie into one single category. Therefore, each issue is placed in its most relevant cell.
The framework as presented in Table 2-2 offers an effective structure in which to identify many pertinent issues of GSD. It highlights that there are many challenges and benefits related to each of the three distances. Furthermore, the literature review has highlighted that there is not full consensus on the opportunities provided by GSD in that it is unclear how they are affected by the challenges of GSD and to what extent they are being realised. These potential benefits are characterised in more detail in the following chapter.

2.15 Researcher’s reflections

This thesis focuses on the realisation of the potential benefits of GSD. Although not each related topic can fit into the framework above, the framework does show that there is a tension between the advantages of developing software in one location against the mainly economic pressures that lead to distributed development.

As previously noted, Herbsleb (2007) discusses the need for distributed teams to build shared knowledge through informal and formal contact. The ‘shared’ knowledge may in fact be interpreted rather differently be different people, depending on the background, experience, prior understanding, and beliefs of each individual involved (Rönkkö 2007). Being such a complex human process, the many challenges related to distributed software development identified in this chapter may not be entirely unexpected.

However, despite the complexities of this collaborative undertaking, the chapter has also identified potential benefits that have been attributed to GSD, many of them being high-level benefits. Again, the promises of cost savings have been often mentioned (e.g. Carmel 2006), along with the flexibility of skilled workforces allowing for fast ramp-up (Herbsleb and Moitra 2001) and follow-the-sun development (Ebert and De Neve 2001). These benefits of GSD can be viewed to be in stark contrast with the knowledge requirements of software development teams (e.g. Rönkkö 2007), where the division of team members by several thousand kilometres and many time zones may be interpreted as insurmountable obstacles to the success of their software development attempts.

Yet, despite these challenges, many small and large companies continue to develop software in a globally distributed context, presumably with an acceptable proportion of those projects being relatively successful. This in turn raises the need to better understand the potential benefits of GSD. Which approaches allow companies to improve their leverage of the unique aspects of GSD? To what extent has the realisation of each benefit been reported, and what are the identified reasons for a benefit not to be realised? In the
following chapter, the potential benefits as identified in the framework are further articulated in order to further understand their implications.
Chapter 3. The Potential Benefits of Global Software Development

3.1 Introduction

The framework presented in Chapter 2 highlights the benefits and challenges for GSD organisations and teams. While not a focus of this research study, resource-based theory could be used to view GSD as a strategic decision in which the challenge lies in matching the organisation’s resources with market needs (Conner 1991b). According to the cited potential benefits in Chapter 2, GSD is indeed a means of making the most of the organisation’s resources in order to meet the needs of its customers by making the software development process cheaper and perhaps faster. However, some cited publications have also pointed to extra drain on the organisation’s resources due to issues such as coordination complexity. Kotlarsky and Oshri (2008) argue that a company considering distributed development should first carefully lay out their strategic goals. Only then can it begin to weigh up the benefits against potential challenges of establishing development activities in any particular location. Being mindful of the research questions laid out in Chapter 1, this chapter seeks to better characterise each of the potential benefits according to existing literature\(^2\). The questions posed in this chapter helped to inspire questioning of interviewees of this case study (which is explained in detail in Chapter 4).

3.2 Potential benefits

DiamondCluster released a report on a survey of the experiences of senior executives with “global IT outsourcing” (DiamondCluster International Inc. 2005). The survey found that the most common benefit of global outsourcing is the flexibility it allows to reallocate high-cost staff to critical business functions. This is made possible by cheaper labour available offshore. The report found that only 37 percent of organisations fully realised their expected benefits of globally outsourcing work, including the benefit of cost savings. Respondents identified expected benefits as: reallocation of internal employees, cost

\(^2\) GSD is a dynamic and evolving phenomenon. Rigorous peer-reviewed published literature may be several years behind state of the art practices and many figures on the trends of GSD come from commercially available reports and reports in the media. As such, this chapter cites some sources that have not been peer-reviewed such as newspaper and magazine articles where their inclusion deemed necessary.
savings, faster time to market, increased shareholder value, and avoiding the need to discontinue or reduce products or services. Approximately 75 percent of respondents agreed that they could achieve cost savings by outsourcing information technology (IT) functions. However, the outsourcing vendors (suppliers) surveyed believed that 95 percent of their clients achieved cost savings because of IT outsourcing. As such, there is a gap between customer and vendor expectations. This points to the problem of setting customer expectations too high before the outsourcing relationship begins. Managing customer expectations in any long-term business relationship is important (DiamondCluster International Inc. 2005). The authors of the survey offered four main reasons as to why the benefits expected from IT outsourcing were only partially realised:

1. Unrealistic or poorly defined expectations established at the outset of the relationship

2. Poorly defined metrics and measures for gauging benefits and success

3. Provider deficiencies in delivering against expectations

4. Buyer deficiencies in managing their providers

With these points in mind, the twelve potential benefits identified in the GSD framework in the previous chapter are further analysed below. According to previous literature, what are the expectations by companies of each of the benefits? To what extent have they been reported to be true, and what challenges may affect their realisation?

To summarise from the previous chapter, the following is a list of twelve potential benefits of GSD according to existing literature, in the order in which they were identified.
Table 3-1. Potential benefits of GSD as identified by the GSD framework.

<table>
<thead>
<tr>
<th>Primary related distances</th>
<th>Benefits</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Improved focus on documentation</td>
<td>(DeLone et al. 2005, Gumm 2007)</td>
</tr>
<tr>
<td>Temporal</td>
<td>Record kept of communications</td>
<td>(Damian and Zowghi 2002)</td>
</tr>
<tr>
<td>Temporal</td>
<td>Improved process definition</td>
<td>(Akmanligila and Palvia 2004, Gumm 2007)</td>
</tr>
<tr>
<td>Temporal</td>
<td>Time zone efficiency</td>
<td>(Espinosa and Carmel 2004, Gumm 2007)</td>
</tr>
<tr>
<td>Geographical</td>
<td>Allocation of roles and team structure</td>
<td>(Ebert and De Neve 2001, DiamondCluster International Inc. 2005)</td>
</tr>
<tr>
<td>Geographical</td>
<td>Increased autonomy</td>
<td>(Gumm 2007)</td>
</tr>
</tbody>
</table>

3.3 Proximity to market and customer

While this benefit has only been briefly covered in the cited GSD literature, it raises the basic question related to all GSD benefits: is it worth the effort to peruse the benefit? For
this benefit, the premise is that the company is targeting customers outside of its own country. GSD is suggested to be an enabler to getting closer to the market and customers by establishing at least parts of the software development work in the target market. Indeed falling trade barriers have facilitated new cross-border GSD activities that may not have been possible in the past (Greenhouse 2003). Meanwhile, existing government regulations may request that some software be developed locally in order to allow a company to participate in that market (Herbsleb et al. 2000).

While presenting a benefit of GSD to practitioners, it raises questions of the effort required to achieve the benefit. Perhaps there are only some contexts that the benefit can be achieved, or that some situations do not allow for the realisation of the benefit. By locating software development activities in the target market, all three GSD distances are introduced. The distances have been identified as causes of challenges to GSD, for example due to their barrier to informal face-to-face communication. Aiming to achieve this benefit may bring the company new local expertise, it would also bring with it challenges.

3.4 Improved focus on documentation

Interpretations of the implementation of software development practices place varying emphasis on documentation through the development activities. Parnas (Ågerfalk and Fitzgerald 2006) has argued (in the face of Agile software development) that documentation has a core role in passing on detailed specifications and other information that cannot be easily transmitted orally. We can take from this that a phenomenon that places greater emphasis on the production of quality documentation is of benefit to the development process as a whole. Of question is whether it is true that GSD contributes to improved quality of documentation without exceeding acceptable levels of effort to create it. The answer to this question is not clear from the findings of previous research.

3.5 Record kept of communications

Damian and Zowghi (2002) cite one advantage of reliance on information communication technologies is that they provide a written record and history of issues being discussed. While this potential benefit has not been cited as a motivating factor of globalising software development efforts, it may be one factor helping globally distributed software development teams. Moreover, it is of question whether this benefit outweighs the
challenges presented by having to communicate through communication technologies rather than in a face-to-face setting.

3.6 Access to large skilled labour pool

This is one of the more important potential benefits of GSD. India, China and Eastern Europe, in particular, have drawn attention due to large numbers of skilled IT professionals in these lower-cost economies (Kobitzsch et al. 2001). Many of the target countries have the benefit of strong educational programs focused at engineering. Of university graduates about 5% of students in the US and Western Europe receive degrees in engineering, compared to roughly 20% in Japan and roughly 40% in China (Carmel and Tija 2005, p. 246). The abundant supply of skilled labour allows an organisation to assign many engineers to one problem, to focus on several projects at one time, and to ramp-up (scale-up) operations within days or weeks instead of months (Carmel and Tija 2005, p. 11).

The population of both India and China stands at over one billion people. Large populations allow these countries to offer sheer economies of scale to potential foreign investors. In 2001 it was stated that India was the world’s second largest pool of English-speaking scientific and technical professionals, with the country boasting a US$5.7 billion software industry rising to an estimated US$50 billion in exports by 2008 (Moitra 2001). In 2001, India had 300,000 people employed in its software industry (Moitra 2001), and there are 180,000 new engineering graduates annually (Vijayan 2003). Moitra added that part of India’s success comes from its world-class quality software development and high process maturity, as has also been stated elsewhere (Vijayan 2003). China is currently best suited for low-level IT work, but hopes to move up the ‘value chain’ in the future (Ulfelder 2003). Its main weakness could be its lack of proficiency in spoken English (Ulfelder 2003). By 2003, China’s IT workforce was 400,000-strong, with 50,000 new graduates annually (Ulfelder 2003).

The massive populations of India’s and China’s IT workforces offer several benefits to organisations practicing GSD. By establishing software development activities there, an organisation can have access to thousands of new employees if skilled IT professionals in the home country may be harder to find. This offers flexibility in ramping up development efforts based on organisational and market demands (Battin et al. 2001, Herbsleb and Moitra 2001). In fact, organisations may not view this only as an up-front benefit, but as a
business necessity where there is competition for highly-skilled staff in the home country (Herbsleb et al. 2000, Nicholson and Sahay 2001).

According to resource-based theory, a firm’s ability to attain and maintain profitable market positions depends on its ability to gain and defend advantageous positions in underlying resources important to production and distribution (Conner 1991a). The resources in question are classified into three categories: physical capital resources, human capital resources and organisational capital resources (Barney 1991). The critical problem faced by a company is how to maintain the distinctiveness of its product – or for identical products, its low cost position – while not investing so much in obtaining these differences as to destroy some normal returns (Cheon et al. 1995). Distinctiveness in the product or low costs are tied directly to the distinctiveness in inputs (resources) used to produce the product (Conner 1991a). Therefore, highly-skilled and lower-cost human capital resources offer increased competitiveness for a company. By efficiently leveraging offshore labour, companies with substantial software development activities should be able to enhance their basic competitiveness. GSD organisations must cost-effectively capitalise on the global resource pool, using scarce resources wherever they are located (Herbsleb and Moitra 2001).

This potential benefit raises the question of how to efficiently leverage large offshore workforces to the ultimate benefit of the organisation. Again, similar challenges face the realisation of this benefit as it introduces geographical, temporal and socio-cultural distance between team members.

3.7 Access to cost-effective labour pool

While cost savings is the primary driving factor of GSD, its cost benefit trade-offs are complex and not well understood (Espinosa and Carmel 2004). Companies attempting to implement GSD strategies soon realise that time zones introduce additional coordination costs that negate some of the labour savings (Carmel 2006).

In a study by Ebert et al. (2001), it was found that co-located teams had 50 percent better defect detection, and dispersion raised the project cost by at least 10 percent. These trade-offs may be ignored in favour of the “instant gratification of savings on wages” (Kharif 2003).
Significantly, little research data have been published to indicate actual cost savings due to GSD. (Companies may have their own data on GSD projects but may be reluctant to make that data available publicly). Extra effort demanded by the GSD context may lead to actual increased monetary costs (Overby 2003b). The costs involved in GSD may not be immediately obvious to companies setting up GSD teams (Kharif 2003). Barthélémy (2001) reported on a survey of 50 companies, highlighting that the benefits of GSD are often eaten away by costs that managers can’t pinpoint: “Companies say they entered an outsourcing agreement believing that they understood all major costs... The costs were not negligible – in some cases, they halved or even cancelled out the company’s potential savings from outsourcing.” Similarly, a survey by Deloitte (2005) found that 38 percent of clients reported paying additional or hidden costs that they originally thought were covered by their contract with the outsourcing vendor. Additional costs may include offshore outsourcing vendor selection, transitioning work to the vendor, layoffs and retention, lost productivity/cultural issues, improving development process, and managing the contract (Overby 2003a).

Cost saving measures can act in themselves as obstacles. As discussed in Chapter 2, publications and literature showed a peak of interest in the potential fear by workers with lower-waged colleagues. The situation discussed here may not be of much relevance to today’s practitioners, although the researcher cannot rule out this being a factor. It had been reported that a company could face a backlash from its own employees if planning to offshore work (Greenhouse 2003, Hayduk 2003, Weakland 2005). American workers in service industries feared for their jobs and for downward pressure on domestic salaries due to offshoring (Venkatraman 2004). An editorial of a special report on offshore outsourcing in Computerworld magazine stated, “I'm prepared for the backlash from daring to run a special report on offshore outsourcing. The O-word always triggers a stream of e-mail invective from displaced IT workers” (Betts 2003). That quote shows how heated debate was in US society regarding jobs being created elsewhere by American-based companies. While attention to this issue seems to have been reduced, it may be still of some relevance.

The survey by DiamondCluster International Inc. (2005) identified risks related to offshoring and outsourcing: employee severance costs, customer reaction, negative publicity, legislation, political pressure, market reaction, union pressure and competitor criticism. A report by Deloitte (2005) states that “real-world experiences suggest that the potential cost savings [from global outsourcing] has been overstated”. Consultancy company Global Insight (USA) published a report stating that cost savings for the entire US IT industry as a result of offshoring in 2003 were US$6.7 billion. This was an “assumed 40 percent savings” versus what would have been spent if domestic resources had been used instead of
offshore resources. The report had projected savings of US$21 billion by in 2008, based on increased overall spending on offshore activities.

Despite the headline figures indicating cost savings as a result of lower salaries, it is unclear from the body of literature whether or not GSD is adequately profitable (when compared to co-located development). It is unclear what the expectations of companies are with regards to cost savings as a result of GSD, and whether those expectations are being met. It remains to be seen whether GSD offers monetary benefits despite added implementation effort. Against this, we should note that indeed cost savings may not always be the primary driving factor for a GSD operation. For example, a company may need to look further afield for skilled employees if there is a skills shortage at its home location. In such a case, the company still needs to examine the factors that may increase the cost of doing business compared to performing only co-located development.

3.8 Cross-site modularisation of work

Cross-site modularisation of work in GSD is a potential benefit, since it enables teams in different geographical locations to coordinate in creating a software system. Admittedly, this proposition borders between being a benefit that GSD offers, and being merely a defining characteristic of GSD. In this thesis, it is treated as a potential benefit as it may be a powerful stepping stone to other benefits of the phenomenon such as providing access to cost-effective multi-skill labour forces.

Good team structure is one of the main facilitators for successful GSD (Cole 2006). The architecture of the system being developed has a major impact on how tasks can be allocated between project teams. Indeed, it is the product’s architecture that should determine the team structure, rather than the other way around (Carmel 1999, p. 127). Effective product architecture can allow globally-dispersed teams to work harmoniously with each other without “stepping on each other’s toes” (Carmel 1999, p. 127). Both Carmel (1999) and Karolak (1998) devote whole chapters of their books to the issue of how to facilitate GSD by effective product architecture and how to allocate tasks across sites.

Good product architecture is heavily based on the concept of modularisation, which is the breaking up of large tasks into smaller components, allowing for work to be carried out on each component in parallel. Carmel (1999) states that clean allocation of tasks is one of the key success factors for globally distributed software development, as it can allow teams to
work at least somewhat independently of each other. Since it can be used to reduce coordination overheads, product architecture is an important coordination mechanism (Herbsleb and Grinter 1999a). More specifically, a system architecture, the planning of (synchronised) project milestones, and the allocation of who does what work where are all vital mechanisms for coordinating work (Herbsleb and Grinter 1999a).

While some publications have stated that task allocation directly impacts on the overall project efficiency (Ebert and De Neve 2001), much earlier work highlighted the importance of focusing on how a product should be structured internally. Conway (1968) made an important realisation that the structure of the system developed reflects the structure of the organisation that developed it. This was the first explicit recognition that the communication patterns of an organisation left an indelible mark on the product that it built (Herbsleb and Grinter 1999b). This later came to be known as Conway’s Law, and Conway stated that it was a necessary consequence of the communication channels between the people doing the work (Conway 1968). In the GSD setting, the modularisation of tasks can therefore shape the communication pathways formed between project members who are globally dispersed.

The relationship between the people working on the system and the product built was also recognised by Parnas (1972). He focused on the process of modular decomposition, and more specifically that a “module” is an assignment of responsibility rather than the explicit part of the software product being built (Parnas 1972). Modular decomposition is the process of dividing the software problem into smaller tractable pieces to work on. Once those pieces (or modules or components) have been defined, and the relationships between them have been made explicit, development work can begin (Grinter 1998). Parnas recognised that the process is not merely a technical division of the product, but a division of labour among individuals. He argued that the advantages of doing a good modular decomposition of the software problem were managerial as well as technical. He said that good modular decomposition would allow for the reduction of communication between developers, as it would eliminate some of the relationships between them. How should the software problem be decomposed? The concept of information hiding was introduced by Parnas (1972). The design concept calls for properly structuring the software’s modules such that the design logic is hidden from its user – the programmer (Carmel 1999). In essence, the less design logic the programmer needs to understand, the fewer the communication needs with other programmers’ modules.
The independence of modules (and therefore the independence of the developers working on them) can be defined by two related concepts: coupling and cohesion. As defined by Carmel (1999), coupling is the degree of interaction between modules, and cohesion is the degree to which a module comprises a well-defined functional whole. Both concepts are at the core of GSD team task allocation. Teams may be defined according to the modularisation of the system being developed. The allocation and location of specific team members can shape the communication patterns across sites. For example, team members who are working on the same module and are located in different sites may need to communicate on a daily basis.

Given that good cross-site modularisation of work may ease coordination issues related to the distances of GSD, it is important to analyse to what extent this may be achieved.

3.9 Improved process definition

From the description of this benefit in Chapter 2, processes may be better defined in distributed projects due to the necessity of optimal common reference points between sites. Investment in a common infrastructure is a prerequisite to this benefit. The premise of this potential benefit is that the entire development process is improved and optimised as a side-effect of overcoming the obstacles of having to coordinate across global distances. However, this raises the question that if co-located software development can get by with less process definition, does the increase of focus on process definition significantly improve the efficiency of the software development.

3.10 Time zone effectiveness

Literature on GSD is contradictory on whether temporal distance offers a challenge or a benefit (Carmel 2006). On one hand, asynchronous communication across temporal distance can allow GSD teams to increase the number of working hours during the 24-hour day. On the other hand, temporal distance has been cited by many as a challenge to communication, coordination and control. According to Carmel (1999), “follow-the-sun” development turns a disadvantage (geographical and temporal distance) into a competitive advantage. Carmel states that the ideal dispersed project spanning multiple time zones can be productive around-the-clock, following the sun as the earth rotates. This is similar to creating 24-hour shifts at one factory. However, Carmel later argued that extreme shifts in work hours is unsustainable since human beings work during the daylight hours due to
deeply embedded biological and societal norms (Carmel 2006). Again, Carmel argued as a key note speaker at the 4th International Conference on Global Software Engineering 2007 for the case of follow-the-sun development after more than a decade of research on GSD. These self-contradictions of Carmel’s publications point to the fact that the question of whether time zone effectiveness is a real benefit of GSD remains unanswered. While this is a potential benefit of GSD, Carmel himself stated in 2007 that not many instances of the practice have been reported (Espinosa et al. 2007).

Follow-the-sun development assumes daily hand-offs from developers in one location who are finishing their work day, to their colleague in another time zone who are just beginning their work day. Developers can be overwhelmed by questions received overnight (Boland and Fitzgerald 2004). Such an operation would require very tight controls of the coordination needed for each work task. Carmel concludes that the number of transitions (or hand offs) required for follow-the-sun is apparently the reason for this approach being still uncommon. In one case study by Carmel (1999), he found that 58% of globally-distributed projects either took longer than normal or there was no difference in software lifecycle time. Follow-the-sun development model also contradicts the best practice offered by Carmel and Agarwal (2001) of reducing intensive interdependence across sites to alleviate challenges of temporal distance.

Espinosa, DeLone and Lee (2006) reported of a company that found that follow-the-sun development was made easier when tasks were loosely-coupled. They note that this can appear to be contradictory, as the follow-the-sun model generally seems to require tight dependency between multiple locations. They found that follow-the-sun was possible when dependencies between tasks could be minimised, or when the management of task dependencies could be simplified and automated as part of the workflow. However, their study does not go into detail about how this could be accomplished.

Part of the confusion over the benefit of time zone effectiveness may arise from the difficulty of breaking apart the effects of geographical distance and temporal distance. The effects of the two distances are not always easily separately identifiable, and much research has covered the both distances, often without distinguishing one from the other (Espinosa and Carmel 2004). Furthermore, temporal distance as a phenomenon is more complex than geographical distance alone (Espinosa and Carmel 2004). Espinosa et al. (2003a) did attempt to distinguish the factors involved.
Overall, the literature remains undecided on whether or not this is a realistic potential benefit. There is an uneasy tension between the advantage of maximising development across the 24-hour day, and the disadvantages that coordinating across temporal distance bring with it.

3.11 Time zone efficiency

Espinosa and Carmel (2004) who proposed this potential benefit also state that a side effect of time zone efficiency is that although coordination costs of direct communication are reduced, costs related to repairing consequences of misunderstandings, reworking, etc., may increase. Overall, while some coordination costs are reduced (tasks are completed by others while you sleep), other coordination costs increase (the effort to coordinate with a remote colleague).

The nuance proposed by Gumm (2007) is that a distributed work setting actually improves team members’ working situation through time zone efficiency. However, in contrast to this, the negative effects of global distances on the software development process have been well documented (see the many challenges to GSD identified in Chapter 2). Overall, the question remains of whether this proposed benefit can be achieved.

3.12 Innovation and shared best practice

The global business environment demands and expects innovative, high-quality software that meets its needs (Highsmith and Cockburn 2001). To stay profitable and competitive, companies usually have to increase their customer value (e.g. by offering innovative products and services) and reduce their project costs and, thus, the price (Lasser and Heiss 2005). Ultimately, organisations aim to globalise their software development to remain competitive (Herbsleb and Moitra 2001).

As Carmel (1999, p. 37) asks, how does one measure the relative innovativeness of global software teams? Naturally, innovation can arise in co-located software teams, and this needs to be compared with the innovativeness of globally dispersed software development teams. Carmel studied certain projects in three companies practicing GSD. One indirect measure of team innovativeness was task enhancement – the degree to which software companies are transferring more of their tasks to the offshore sites, particularly high-level, value-add tasks. Carmel (1999, p. 37) found that in 69% of cases, the offshore sites were
either assuming high-level design responsibilities or even full product ownership. Product ownership is taking responsibility for the release’s definition all the way through design and development. In 47% of the cases, the offshore teams were working on the companies’ flagship products, and in some cases, the offshore site initiated (“visioned”) the product itself and then went on to design it.

Innovation and sharing of best practices may come with the maturation of those sites involved. Originally used simply as an offshoring destination, Ireland has proven itself to be an innovative destination, evolving into a software development hub that taps into other offshoring markets (Holmström Olsson et al. 2008, Richardson et al. 2008). With Ireland only being one example of a country being heavily involved in GSD, we can speculate that, given time, any country involved can evolve to become a hub of further innovation.

3.13 Allocation of roles and team structure

The premise of this potential benefit is that employees in the company’s established sites remain employed while individuals with cheaper salaries also join the organisation. The organisation would also need to be adaptable in its approach in order to welcome these new employees into existing projects and control structures. The details of this potential benefit need to be further investigated.

3.14 Increased autonomy

This potential characteristic of GSD has only been branded by Gumm (2007) as a benefit. While it may indeed be beneficial for teams to be able to establish independent working practices, it is again questionable whether this potential benefit can outweigh the costs of having to coordinate across distances. Hence, similar questions arise as with other benefits that have already been analysed above.

3.15 Summary

This chapter has attempted to further clarify the twelve potential benefits of GSD that were identified in the GSD framework in the previous chapter. This includes the main motivating factors of GSD such as the benefits of proximity to market and customer, access to large cost-effective labour pools, and follow-the-sun development. There are
indications that not all benefits are realised by all projects, and there is not agreement on whether or not all potential benefits can be achieved. Therefore, it is necessary to better understand each of the benefits how they can be achieved. The potential benefits of GSD identified in this chapter serve as a basis for the empirical study described in the following chapters.
Chapter 4. Research Methodology

4.1 Introduction

This chapter deals with how this empirical study was planned and implemented. First the philosophical underpinnings of the empirical study are set out. Second, the research methodology and method employed by the study are examined in contrast with other possible research methods. Third, the research design decisions required for employing an interpretive qualitative multiple-case study are laid out. Finally, an account of how the research was implemented is given, including an introduction to the two companies involved. All of the activities described in this chapter have been carried out with the fundamental goal of satisfying the research objective from Chapter 1, and to address the resultant research questions.

4.2 Interpretive study

The study described in this thesis was an interpretive one. This statement indicates that the researcher took an interpretive approach towards understanding the phenomenon, thus defining how the subject was approached, and how data were obtained and interpreted. The phenomenon was approached as a social one that is interpreted in the minds of participants; the phenomenon could not be understood without understanding how participants make sense of the world.

All research (whether it is quantitative or qualitative) is based on some underlying assumptions about what constitutes ‘valid’ research and which research methods are appropriate (Myers 1997). The most fundamental set of assumptions held by a professional community then allows its members to share similar perceptions and engage in commonly shared practices, which, according to Hirschheim and Klein (1989), is called a “paradigm”. As Chua (1986, p. 604) puts it:

“Given this mutually interactive coupling between knowledge and the human, physical world, the production of knowledge is circumscribed by man-made rules or beliefs which define the domains
While Chua’s domain was accounting, his points were still very much valid for information systems research (cf. Orlikowski and Baroudi 1991). A paradigm is made up mainly of two different types of assumptions:

1. Assumptions about the world (ontological)

2. Assumptions about knowledge (epistemological)

**Ontological** beliefs relate to the nature of the phenomenon itself – whether the world is assumed to be objective (independent of humans), or subjective (having existence only through humans creating and recreating it) (Orlikowski and Baroudi 1991). **Epistemological** assumptions dictate what is an acceptable truth, by specifying the criteria and process of assessing truth claims (Chua 1986). For example, an epistemological assumption may allow for a theory to be considered true if empirical research cannot falsify it. In relation to this, methodological assumptions dictate what types of research approaches are valid for collecting valid empirical evidence. Finally, a research paradigm also constitutes the beliefs about the relationship between knowledge and the empirical world (Chua 1986, Orlikowski and Baroudi 1991); that is, the role of theory in the empirical world, and how the values and intentions of the researcher relate to the phenomenon.

Orlikowski and Baroudi (1991), following a classification by Chua (1986), suggest three categories of research epistemologies found in information systems research: positivist, interpretive and critical. Again, a research epistemology refers to the assumptions about knowledge and how it can be obtained (Hirschheim 1992). As such, research studies classified as positivist will draw on a different set of beliefs and assumptions to that of interpretive studies. The following section discusses the underlying assumptions of this interpretive study, given its interpretive stance.

### 4.2.1 Interpretivism – ontology and epistemology

This interpretive study of GSD – interpretivism being a “soft” approach (Fitzgerald and Howcroft 1998) – fundamentally assumed that most of our knowledge is gained through social construction such as language, consciousness, shared meanings, documents, tools
and other artefacts (Walsham 1995b, Holmström 2004). The understanding of human thought and action in social and organisational contexts is of primary interest in interpretive research (Klein and Myers 1999).

The phenomenon at the focus of this study (GSD, and specifically, its potential benefits) was subjective to the practitioners involved in the study. The study relied heavily on the understanding of social and organisational contexts and the work processes involved. This meant that research methods considered inclined more towards contextually dependent observational approaches, rather than experimental approaches, which is in line with the argument given by Davies and Nielsen (1992, p. 181). It involved detailed qualitative research, where an interpretive approach was seen as a suitable approach, and where a hypothesis was not being tested.

Interpretive research does not predefine dependent and independent variables, but rather focuses on the complexity of human sense making (Kaplan and Maxwell 1994). As such, interpretive studies generally attempt to understand the phenomenon through the meanings that people assign to it. Interpretive researchers believe that social reality can only be interpreted, and not “discovered” as in the positivist tradition (Orlikowski and Baroudi 1991). This is in direct contrast to positivist research where there may be an “objective” or “factual” universal account of events and situations – the interpretive researcher instead seeks a relativistic, albeit shared, understanding of the phenomenon (Orlikowski and Baroudi 1991). Organisations, groups, or social systems do not exist apart from humans, and hence cannot be apprehended, characterised, and measured in some objective or universal way. Interpretive research is based on the epistemological belief that social process is not captured in hypothetical deductions, but that understanding social process involves getting inside the world of those generating it (Rosen 1991).

Interpretive studies in information systems (IS) are “...aimed at producing an understanding of the context of the information system, and the process whereby the information system influences and is influenced by the context” (Walsham 1993). This philosophy is receiving increased attention and popularity in many social science fields (Orlikowski and Baroudi 1991), and has emerged as an important strand in information systems research. Interpretive research has

3 It cannot be overlooked that the researcher worked within a research group whose topical interests leaned towards the interpretive tradition. While being in no part a justification for having carried out an interpretive study, it must be made clear that this approach was strongly considered due to this background.
the potential to produce deep insights into information systems phenomena including the
management and development of them (Klein and Myers 1999).

Orlikowski and Baroudi (1991) classified research as interpretive where there was evidence
of a nondeterministic perspective with the intent of the research being to increase
understanding of the phenomenon within cultural and contextual situations; where the
phenomenon of interest was studied in its natural setting (as opposed to settings of
controlled scientific experimentation); and where researchers did not impose their
outsiders’ a priori understanding of the situation.

The interpretive research tradition has been subject to criticism. These criticisms or
limitations of an interpretive view were taken into account by the researcher when forming
the empirical study. Despite these criticisms, the interpretive approach was deemed highly
suitable for the study of the phenomenon. Fay (1987) suggests four deficiencies of the
interpretive perspective:

- The interpretive perspective does not examine the conditions, which are often
  external, which give rise to certain meanings and experiences.

- It omits to explain the unintended consequences of action, which often help shape
  social reality.

- It does not address structural conflicts within society and organisations, such as in
  situations where participants’ accounts of action and intentions are inconsistent
  with their actual behaviour.

- It neglects to explain historical change: how the particular social order came to be,
  and how it may change in the future.

With respect to the relationship between theory and practice, the interpretive researcher
can never assume a value-neutral stance, and is always implicated in the phenomena being
studied (Orlikowski and Baroudi 1991). The research will always be somewhat affected by
the researcher’s prior assumptions, beliefs, values, and interests, and the researcher must
account for this. From that, there are two existing views on the role of the researcher. In
the “weak” constructionist view, the researcher aims to understand the existing meaning
systems shared by actors, and interprets them in the analysis of the research. In the
“strong” constructionist view, the researcher should not only describe the phenomenon in the words of the actors, but also be presumed to enact the social reality she is studying. The researcher in part creates the reality she is studying. The empirical study described in this thesis can therefore be labelled as coming from a “weak” constructionist view, as the researcher (while aware of his existing biases towards the general phenomenon) did not attempt to participate in the phenomenon being studied, but rather sought to understand the interpretations of the practitioners.

The following sections discuss the alternative world views of positivism and critical research, followed by a discussion of the popularity of research approaches in information systems.

4.2.2 Positivism

It is of the researcher’s view that a positivist study on the phenomenon of GSD would have also been valid, but that such a study would not take into account the social and organisational context involved, and would indeed arrive upon a different set of findings. For examples of such work, see Herbsleb et al.’s studies which are based on measurements and statistical data from sources such as a change management system and an instant messaging server to analyse the nature of globally-distributed software development (Herbsleb et al. 2002, Herbsleb and Mockus 2003). The merits of taking an interpretive view of the phenomenon have been described in the previous sections.

Positivist research studies – positivism being a “hard” approach (Fitzgerald and Howcroft 1998) – generally assume that reality is objectively given and can be described by measureable properties which are independent of the observer (researcher) and their instruments (Myers 1997). The positivist research tradition has its roots in the natural sciences. The studies assume the existence of fixed relationships within the phenomena which are typically investigated with structured instrumentation (Orlikowski and Baroudi 1991). Positivism has been criticised for its emphasis on cause-effect relationships and its deterministic view (Fitzgerald and Howcroft 1998). In their review of existing IS literature, Orlikowski and Baroudi (1991) classified studies as positivist where there was evidence of formal propositions, quantifiable measures of variables, hypotheses testing, and the drawing of inferences about a phenomenon from the sample to a stated population. Ontologically, positivist researchers assume that the objective and social world exists
independently of humans, and whose nature can be relatively easily understood, characterised, and measured (Orlikowski and Baroudi 1991). The role of the researcher is to “discover” the objective physical and social reality by precisely measuring the dimensions of interest to the research. The researcher is himself/herself detached from the phenomenon of interest. It is the positivist researcher’s belief that there exists a theory-independent set of observation statements that could be used to confirm or verify the truth of a theory (Chua 1986, p. 607). This approach has two main consequences (Chua 1986):

- It leads to the search of universal laws or principles from which lower-level hypotheses may be deducted.
- There is a tight linkage between explanation, prediction, and technical control.

4.2.3 Critical

Critical researchers assume that social reality is historically constituted and that it is produced and reproduced by people (Myers 1997). While people can consciously try to change their social and economic circumstances, critical researchers acknowledge that their ability to do so is limited by various forms of social, cultural and political constraints (Orlikowski and Baroudi 1991, Myers 1997). Critical studies aim to critique the 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 conditions (Orlikowski and Baroudi 1991). Critical research, more so than the other approaches, attempts to critically evaluate and transform the social reality under investigation. Such studies have an objective of creating awareness and understanding of the various forms of social domination, so that people can act to eliminate them. As such, this is the researcher’s role regarding the relationship between theory and practice. The criteria used by Orlikowski and Baroudi (1991) to classify studies as critical were:

“evidence of a critical stance towards taken-for-granted assumptions about organizations and information systems, and a dialectical analysis which attempted to reveal the historical, ideological, and contradictory nature of existing social practices.”
4.2.4 Discussion

From their survey of 155 IS research articles published in four major US outlets between 1981 and 1988, Orlikowski and Baroudi (1991) classified 96.8 percent of them as positivist, 3.2 percent of them as interpretive, and none of them as critical. Chen and Hirschheim (2004) extended the work of Orlikowski and Baroudi by examining paradigmatic and methodological aspects of IS publications from 1991 to 2001 in four US and four European major publication outlets. They found that 81 percent of the studies followed the positivist tradition, and 19 percent followed the interpretive tradition. A geographical analysis found that the US journals, as opposed to the European journals, tended to be more positivist.

According to Orlikowski and Baroudi, the strong dominance of the positivist tradition, by not acknowledging the legitimacy of other research traditions, has limited what aspects of IS have been researched, and how they have been studied. Indeed Walsham (1995a) implies that interpretivism was not being seriously considered (or was not at all considered) by main-stream journals until the early- to mid-1990s. In a later publication Walsham (2006) states that interpretive research in IS is now a well-established part of the field. Walsham cites Curtis et al. (1988) as an important interpretive study in the IS field, where they reported on interviews with 17 large software projects. Furthermore, Fitzgerald and Howcroft (1998) argue the futility of “extremist” research views between “soft” and “hard” approaches. There has been a polarisation, they argue, of the two sets of approaches, with both sides claiming superiority over the other. They go on to say that there is no approach proven to be universally acceptable, and so the debate cannot be solved in this way. Both approaches do have their own strengths and limitations, and therefore their application should be chosen according to their appropriateness to the research question at hand.

Table 4-1 provides a summary of the research epistemologies that have been covered in this section, and how they interpret reality and its connection to knowledge. The table is based upon the categorisation by Orlikowski and Baroudi (1991) and draws from the comparison of dichotomies provided by Fitzgerald and Howcroft (1998, p. 323). A similar set of comparisons is also offered by Chua (1986).
Table 4-1. Summary of research epistemologies.

<table>
<thead>
<tr>
<th>Reality</th>
<th>Positivist</th>
<th>Interpretive</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Objective physical and social world exists independently of a person's cognition.</td>
<td>• Understand phenomena through meanings that people assign to them.</td>
<td>• Reality constrained by social, cultural and political factors.</td>
</tr>
<tr>
<td>Knowledge</td>
<td>• Concerned with empirical testability of theories – “verified” or “falsified” theory.</td>
<td>• Knowledge gained through social construction.</td>
<td>• Constraints brought to light.</td>
</tr>
<tr>
<td></td>
<td>• No universal truth.</td>
<td>• No universal truth.</td>
<td>• Attempts to critically evaluate and transform social reality under investigation.</td>
</tr>
<tr>
<td>Relationship between theory and practice</td>
<td>• As impartial observers, researchers can objectively evaluate or predict actions or processes.</td>
<td>• Researcher is naturally biased coming into the research.</td>
<td>• Researcher’s role is to bring constraints to light, to help change the social reality.</td>
</tr>
<tr>
<td></td>
<td>• Researcher detached from the phenomenon.</td>
<td>• “Weak” and “strong” constructionist views.</td>
<td></td>
</tr>
</tbody>
</table>

4.3 Research Methodology

The underlying research epistemology of a research study influences its research methodology, which in turn influences which research methods may be selected (Myers 1997). For this interpretive study, the researcher adopted a qualitative research methodology, and the empirical study therefore dealt with the collection and analysis of qualitative data.

Silverman (2006, p. 15) describes four components of a research methodology:

1. A preference for certain research methods.
2. A set of assumptions about the nature of reality, and the role of the researcher in it.
3. A range of solutions for addressing a research problem.
4. A systematic sequence of procedural steps for a given research method.
A research methodology encompasses the choices and decisions made concerning a research project (Silverman 2006, pp. 13-15). It covers how the researcher plans and implements their research project, including how data is collected and analysed. In broad terms, research can be classified along two methodological definitions: qualitative and quantitative research.

Whereas quantitative research uses mathematical and statistical techniques to measure and analyse the data, qualitative data, however, makes use of thick description to determine “what things exist rather than how many there are” (Fitzgerald and Howcroft 1998). Given the interpretive stance of this study, it is taken that knowledge is gained through social construction such as language, consciousness, shared meanings, documents, tools and other artefacts. The ultimate goal of this study was to deepen our understanding of the markedly social phenomenon of GSD. It was decided that qualitative data would be ideal in helping to better understand the phenomenon given the thick descriptions that could be formed from these social constructions, through the interpretations expressed by the participants.

In IS, there has been a shift from technological to managerial and organisational areas, resulting in greater interest in qualitative research methods (Myers 1997). In fact, there is a widely held opinion in IS research that the scientific method of inquiry is inappropriate and limiting (Myers 1997). The interpretive perspective has been linked with the qualitative methodology, however, this is not a requirement (Walsham 1995a). A qualitative study can be strongly positivist in its explanation of social events. Similarly, a research method may be appropriate for more than one research epistemology. For example, a case study can be positivist (Yin 2003b), interpretive (Walsham 1995b), or critical.

4.3.1 Research methods

As already stated, the research methodology applied in an empirical study will tend to define the subset of acceptable research methods for that study (Yin 2003b, p. 3). Indeed, for this interpretive qualitative study, the case study approach was the research method which was selected to enable the collection and analysis of appropriate qualitative data. The selection of that research method is described in the next section.
A research method is a specific research technique (Silverman 2006). Each methodology (as Yin terms a “strategy”) is a different way of collecting and analysing empirical evidence, following its own logic (Yin 2003b). By applying an appropriate research method, relevant empirical data may be collected. A qualitative approach seeks to collect qualitative data through methods such as interviews, review of documentation, and participant observation data which can help the researcher understand and explain social phenomena.

A variety of well-documented research methods exist (Marshall and Rossman 1999, p. 60). There is a considerable range of methods available to the information systems researcher (Galliers 1992), all of which have their strengths and weaknesses. Moreover, no single research method is universally applicable (Gill and Johnson 1991). The choice of method is determined by a number of factors including: goodness-to-fit with the research objectives, past experience of using methods, whether the project is a solo or collaborative effort, time constraints, availability of funding, the researcher’s philosophical beliefs, the political context of the research, the nature of the phenomenon being studied, and access to suitable sources of primary and secondary data (Mumford 1985, Gill and Johnson 1991, Yin 2003b).

Several publications have analysed the number of published studies in respected IS journals according to which research methods were applied. See Table 4-2 for the findings of the literature surveys by Orlikowski and Baroudi (1991), and which was later extended by Chen and Hirschheim (2004).
Table 4-2. Popularity of research methods in IS.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Survey</td>
<td>49%</td>
<td>41%</td>
</tr>
<tr>
<td>Laboratory Experiment</td>
<td>27%</td>
<td>18%</td>
</tr>
<tr>
<td>Case Study</td>
<td>14%</td>
<td>36%</td>
</tr>
<tr>
<td>Mixed Method</td>
<td>3%</td>
<td>-</td>
</tr>
<tr>
<td>Field Experiment</td>
<td>3%</td>
<td>2%</td>
</tr>
<tr>
<td>Instrument Development</td>
<td>3%</td>
<td>-</td>
</tr>
<tr>
<td>Protocol Analysis</td>
<td>1%</td>
<td>-</td>
</tr>
<tr>
<td>Action Research</td>
<td>1%</td>
<td>3%</td>
</tr>
</tbody>
</table>

Chen and Hirschheim found that survey and case studies were “by far the most popular” research methods. They note the rise of popularity of case study research, the number of which actually exceeded the number of survey studies in 1997, 1998 and 2001. They speculate that such results might indicate that researchers are not just engaging in more empirical studies but rather paying more attention to how the phenomenon evolves. The following section describes the choice of case study as this study’s research method.

4.3.2 Interpretative qualitative case study

As a summary, a case study research method was deemed to be the most suitable strategy of approaching this interpretive qualitative empirical study for several reasons. First, case study research is suitable for investigation of social and organisational phenomena. It was deemed to be a good fit for the investigation of the social and organisational phenomenon of GSD. Studies focusing on society and culture, whether a group, a program, or an organisation, typically espouse some form of case study as an overall strategy (Marshall and Rossman 1999, p. 61). Second, case study research can lead to valuable thick description that can help to further our understanding of the phenomenon (Yin 2003b). Thirdly, the approach can be executed without spending extended periods of time at the case site (as opposed to months or years typically spent in ethnographical studies). The approach allows the researcher to elicit interpretive qualitative data from participants while not participating directly in the phenomenon.
The case study approach for this empirical study was based on the textbook by Yin (2003b). The selection of this approach was taken on the understanding that although his work is a primer in case study research, it is also, according to Dutton (1988):

"an excellent introductory text that developed guidelines for conducting case research. Yin offers much advice to those approaching case study research for experienced as well as novice researchers."

While Yin (2003b) takes a positivist stance, Wang (2007) states that “[Yin’s] view that case study is the preferred research strategy to answer ‘how’ and ‘why’ questions would also be accepted by the interpretive school”. While the case study described in this thesis examined qualitative data, a case study may be qualitative or quantitative, or both (Huberman and Miles 2002, p. 9).

The case study method is compatible with the interpretive world view in that it allows for the researcher to analyse the meanings given to phenomena by the actors by studying the actors in their typical work settings as opposed to controlled environments for scientific experimentation. In general, case studies are the preferred strategy when ‘how’ or ‘why’ questions are being posed, when the investigator has little control over events, and when the focus is on a contemporary phenomenon within some real-life context (Yin 2003b). Therefore, the case study method is helpful in situations where the researcher is seeking to develop understandings of the dynamics of the phenomenon in its natural context (Yin 2003b). These characteristics made the case study approach very suitable for this empirical study. The researcher was aware that given his access to participant companies, he would not have control over events in the organisations, but rather, could facilitate practitioners in reflecting on the phenomenon in order to elicit qualitative data.

Yin (2003b) presents the case study approach as a comprehensive research strategy, rather than a method that only covers data collection, or is merely a design feature. Yin (2003b, pp. 13-14) defines the case study approach in two parts. First, it is an empirical inquiry that “investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly defined”. Second, the enquiry copes with many variables of interest, it relies on multiple sources of evidence without needing to converge in a triangulating fashion, and it benefits from prior development of theoretical propositions to guide data collection and analysis.

Case studies can include either single or multiple cases, and numerous levels of analysis (Yin 2003b). This empirical study was a multiple-case study, the implications of which are
discussed later in this chapter. In the sub-sections that follow here, it is explained why the case study approach was deemed to be the most suitable amongst research methods in order to pursue answers to the research questions.

4.3.3 Alternative Research Methods

Case study research has an emphasis on context, with an ability to develop a better understanding through the collection of “thick” description of the phenomenon of interest (Yin 2003b). In contrast, the survey method does not lend itself to such outcomes. As Whitely (1998) puts is:

“Investigating what developers do ‘in the wild’ is not something that can be adequately addressed by large scale postal surveys or laboratory experiments. Instead, what is required is a more qualitative approach which is able to pick up the nuances of the situation.”

Case study research can be distinguished from other research methods. For example, research involving experiments (and especially laboratory experiments) which remove the phenomenon from its context in order to control the number of variables. Surveys may attempt to deal with the context, but their ability to do so is extremely limited. The survey designer must limit the number of questions so as to attract an acceptable number of responses.

As previously described, a variety of factors help determine whether or not a certain research method is suitable for the research project. The primary factors are the research topic itself, and the research questions. Flow diagrams of the ‘standard’ process often depict the research method of being consequential to the objectives set out (Lang 2006, p. 80), however this may not fully match events in reality. Lang (2006) sets out several reasons for this. First, it may be almost impossible to plan what needs to be done without knowing in some way how that needs to be accomplished. Second, when designing the research study, it is essential for the researcher to think ahead and consider such factors as time and resource constraints, and access issues. This limits the choices of the research method that may be adopted. As such, Lang states that “the determination of research questions is therefore not an entirely open and independent decision, but a pragmatic one which should anticipate the range of feasible research methods available.”
When attempting to address the research questions taken from Chapter 1, the researcher was aware that he would not have sufficient access to companies to be able to spend extended periods there. In this case, an ethnographic study was ruled out, as this would require a high degree of access to the development teams over an extended time period. Likewise, action research was also ruled out. Field and laboratory experiments were also ruled out because of their incompatibility with the epistemological standpoint at the core of this research. It was accepted that direct access could only involve elicitation techniques such as interviews and formal meetings. This lead to a choice involving survey and case study methods – either adopting one of these methods, or applying a combination of both. The merits of the case study method and its compatibility with the research questions lead the researcher to adopt this method of research specifically.

Survey

By a survey, it is meant a questionnaire administered in person, by mail, over the World Wide Web, or in other formats (Lang 2006). The survey method has precise procedure which, when applied appropriately, yields valid and easily interpretable data (Pinsonneault and Kraemer 1993). Surveys can produce broad descriptions and enumerations of the study population, and can provide the opportunity for statistical analysis and broad generalisability (Dillman 2000). The survey approach provides the researcher with a collection of “snapshot” data about the characteristics and opinions of a large population, can be used to statistically test a formal hypotheses, and is appropriate for both exploratory and descriptive research (Lang 2006).

Traditionally, large-scale surveys have been operated as postal surveys, but online access provided researchers with the potential of web-based surveys (Lang 2002). Many Web-based surveys are published under the assumption that they can quickly and easily generate high numbers of quality responses. However, Lang argues that, as with all other modes of survey research, good quality responses cannot be expected except where careful attention has been given to sampling procedures and other critical aspects of research design. Lang suggests that Web-based surveys can be strengthened if included in a multi-mode research design, including surveys by mail and by e-mail. Pinsonneault and Kraemer (1993) found in their survey of literature that the survey method had often been misapplied and suffered from five major weaknesses:

1. A single method being used where more would have been appropriate.
2. Inadequate sampling procedures.

3. Low response rates.

4. Incoherent linking between the unit of analysis, and the respondents.

5. Heavy reliance on cross-sectional surveys, while there was a lack of longitudinal studies.

Laboratory experiment

The premise of a laboratory experiment is that the environment of the study can be controlled allowing for replications of the initial study (Miller 2005). In a quantitative approach, laboratory experiments identify precise relationships between chosen variables using analytical techniques (Braa and Vidgen 1999). Given the level of complexity in the GSD phenomenon, it may be very difficult to describe through a finite set of variables; it may not be possible to describe all of the facts regardless of the underlying framework (Miller 2005). To that effect, Miller argues for a range of acceptable approaches to replication, making it possible to choose the most appropriate design characteristics for the replication. To paraphrase Braa and Vidgen (1999), a laboratory experiment approach takes place within an unrealistic controlled environment. This is the biggest fault seen in a laboratory experiment were it to be applied to the phenomenon of GSD. Indeed, they argue that for IS research, the primary “laboratory” where technical artefacts can be studied in-context is the organisation itself. A laboratory experiment does not generally allow for an in-context study.

Field experiment

The field experiment approach is an extension to the laboratory approach, bringing it into an organisational context (Braa and Vidgen 1999). A field experiment is by nature less rigorous than a laboratory experiment, as there are many factors which cannot be controlled by the researcher but which may still affect the outcome. The field experiment approach attempts to make generalisable experiments while still observing the phenomenon in its natural context. While the real-life setting is advantageous, Braa and Vidgen point out some weaknesses to this approach:
• It is difficult to find organisations that are prepared to be (rigorously) experimented on.

• Replication issues are prevalent, as it is very difficult to sufficiently control the environment such that the study may be replicated in a valid manner afterwards.

**Action research**

Action research was developed during the 1950s in order to study social psychology within the framework of field theory (Baskerville and Wood-Harper 1998). The approach was not adopted for this research study, as it demands close cooperation and facilitation by the organisation. The approach is generally divided into five phases which are iterated through cyclically: 1) diagnosing, 2) action planning, 3) action taking, 4) evaluation, and 5) specifying learning.

Baskerville and Wood-Harper (1998) point towards several reasons for concern involving action research, such as: action research is sometimes branded as ‘consulting masquerading as research’, and that it is difficult to determine the cause of a particular effect that could be due to the environment, researcher, or methodology. It is an interventionist approach since actual changes need to be applied to the organisation, with the effects of the changes then under study.

**Ethnography**

Ethnographic research originates from social and cultural anthropology where a researcher spends significant time in the field (Myers 1999). Ethnographers immerse themselves in the lives of the subjects (Lewis 1985), and aim for a holistic view of the social and cultural context. It is therefore one of the most in-depth research methods available to researchers (Myers 1999). For this immersive view of the phenomenon, the researcher is required to spend an extended period of time studying the phenomenon within its natural context. Similar to the case study approach, it can give rich insights into understanding people, the organisation, and how people interact within their environment.

Indeed, the main up-front difference between case study and ethnographic studies is the amount of time the researcher needs to spend immersed in the context (Myers 1999). The element of time is a strength of ethnography, as it allows the researcher to study the subjects in their normal or natural environment. This allows the researcher to see the
phenomenon through the actor’s eyes. Rather than studying the accounts or participants, an ethnographic approach gives the researcher first-hand experience of the phenomenon being studied. It is for the practical reason of time constraint that this approach was not embarked upon by the researcher for this empirical study.

4.4 Research Design

The following sub-sections describe the design decisions taken during this interpretive case study research. A research design is a “blueprint” for the research study covering what questions to investigate, what data are relevant, what data to collect and how to analyse the results (Philliber et al. 1980). It helps to guide the research study so as to avoid a situation where the evidence collected does not address the research questions (Yin 2003b, p. 21). When designing a research study, several things must be initially considered, including the type of study, the logic of the study, single or multi-case designs, unit of analysis, data collection techniques, and case report scheme (Wang 2007). The following sub-sections attempt to cover those topics in relation to the practicalities of the case study.

Dutton (1988) argues against accepting one particular reconstruction of the logic of case study research as a definitive account of the logic of case study research, namely that of Yin (2003b). Like Yin, Walsham (1995b) has provided an important contribution by describing the nature and method for interpretive case studies. He states that there is a lack of “a synthesised view of the nature and conduct of case studies with specific reference to the field of computer-based IS”, and the volume and range of such studies have been limited. It should be noted that the researcher experienced a learning curve in interpreting the intricacies of applying the case study research method in practice. Throughout the planning and implementation of the study, the intended design of the research study was reviewed and revised to best match the research strategies laid out by Yin and Walsham.

4.4.1 Exploratory case study

This case study was exploratory in nature. The research of GSD is an established research area. Some work has already contributed to the theorising of the phenomenon (for example, Espinosa and Carmel (2004) present a dyad model concerning the effect of temporal distance on coordination costs). However, there have been relatively few
published empirical studies focusing on the realisation of its potential benefits. An exploratory approach was deemed to be suitable to the research objectives as laid out in Chapter 1 in order to come to a deeper and more nuanced understanding of the factors involved.

Exploratory studies are known as “inductive” or “theory-building” research (Eisenhardt 1989). In such studies, the researcher does not begin with an established theory. Rather, a research question is defined in order to give focus to the study. The researcher may also begin with a specification of constructs. Eisenhardt sees this step as valuable as the constructs may lead to a firmer empirical grounding for the study. This research study followed this advice, in that the GSD framework presented in Chapter 2 was used for the multiple-case study so as to give foundational constructs to understand the phenomenon. In effect, the use of the framework to structure the approach to investigating the research questions allowed the researcher to ‘bootstrap’ the investigation, giving it structure and direction. While not being a fully-formed theory, the theoretical concepts of the framework helped the case study in several ways, as Yin (2003b) describes:

- Selecting the cases to be studied, whether to study single case or multiple cases.
- Specifying what is to be explored in exploratory case studies.
- Defining a complete and appropriate description when doing descriptive case studies.
- Stipulating rival theories when doing explanatory case studies.
- Generalising the results to other cases.

Theory development prior to collecting data is one differentiating factor between case studies (Van Maanen 1988) and grounded theory (Strauss and Corbin 1998). In support of the use of the GSD framework during the research design process, Yin argues that for case studies, theory development as part of the design phase is indeed essential, whether the case study’s purpose is to develop or test theory (Yin 1994, p. 27). Walsham (1995b) states that the use of theory in interpretive research early on in the process allows for the creation of an initial theoretical framework which takes account for previous knowledge that has been published by other researchers. Walsham also argues that the initial use of theory
represents a danger to the integrity of the research process in that it may be applied too stringently, thus limiting the researcher in forming a holistic view of the phenomenon. While the researcher was mindful of this fact, he strived towards investigating the research questions through the exploratory nature of the case study asking participant ‘how’ and ‘why’-type questions to reveal their interpretations of the phenomenon.

An outcome of the exploratory research was a ‘thick description’ of the phenomenon. The concept of thick description comes from the anthropological tradition of research (Walsham 1995b). The IS researcher is faced with a “multiplicity of complex conceptual structures” coming from the investigation of complex organisational phenomena. The process of creating a thick description of the phenomenon is important for the understanding of what is happening within in.

4.4.2 Single and multiple cases

Of course, a research case study involves the study of at least one case. The definition of what the ‘case’ encompasses is described in the next section with regard to defining the unit of analysis. The design of a case study design must address whether the study will include a single case or multiple cases; each design is appropriate under several circumstances (Yin 2003b, pp. 39-53). A single case study can be used in a deductive approach if dealing with the “critical case”, where that case matches the set of propositions and circumstances believed to be true according to the underlying theory. The single case study is suitable for the “extreme” or “unique” case. For example, Carmel (2006) reports on a single case study on the large outsourcing vendor Infosys. The study highlights the practices of temporal flexibility by Infosys employees, yet Carmel draws conclusions for GSD as a whole. Conversely, a single case can be a “typical” case where the aim is to study an everyday or commonplace situation.

This empirical study was designed to include data from multiple organisations practicing GSD. Such an approach is termed “multiple-case design” (Yin 2003b). Multiple-case studies are often considered more compelling, and the overall study may therefore be regarded as more robust (Herriot and Firestone 1983, as cited by Yin 2003b). Yin argues that the reason that multiple-case studies are preferable is because the analysis can compare and contrast two or more cases. Analytic conclusions arising independently from two cases can be more powerful than a single case, especially if similar conclusions arise from
multiple cases, each with its own context. Two cases may also be chosen because of their contrasting situations.

4.4.3 Unit of analysis

The unit of analysis defines what the “case” is for a case study (Yin 2003a, p. 3). The case may be a person, an event, or an entity that is less well defined than a single individual. For example, Lang (2006) conducted his field study comprised of individuals from fifteen different organisations, with the focus being on the experiences of the individual, while the context was their organisation. Taking Carmel (2006) as an example, the focus of the case was the Bangalore campus of Infosys, while almost assuming that this applies directly to the company as a whole. In the context of software development, a team may be a logical unit of analysis. Teams have long been defined partly in terms of stable boundaries and membership (Alderfer and Smith 1982), yet teams’ boundaries are often permeable (Ancona et al. 2002). Their members can often change (Arrow and McGrath 1993). A company division leading the development of a software product may also present a relevant unit of analysis, being made up of multiple teams. Furthermore, in the context of GSD, Lang (2006) indicates that a case may be defined as a single site or location of a development team, if that development team was distributed across several locations. In her study, Kirsch (2004) defined the smallest unit of analysis to be in fact one development phase of a project.

A case study involving only one case may still have more than one unit of analysis (Yin 2003b, pp. 42-45). A holistic view of the case may be taken, or multiple embedded units may be identified within the same overall case. Given the relatively nascent state of the GSD literature defining the potential benefits, and the exploratory nature of this study, the researcher found it most suitable to take the software development division of each company to be the unit of analysis. If this had been an explanatory study seeking to test a specific theory, it would have been probably more suitable to take a specific project or team as the unit of analysis, and to test the theory within that team. The rational for this unit of analysis design decision was based on several points:

- It allowed for data to be collected without being limited to the interviewee’s current assigned team or project. This allowed for a more expansive and holistic view of each organisation. Interviewees were able to relate topics of discussion with
experiences of previous projects and teams within the same organisation, where appropriate.

- Setting the software development division as the unit of analysis allowed for a broader range of data rather than restricting the data collection to one team specifically.

- The researcher acknowledges that having taken the design decision, the case study sites of both companies were not necessarily representative of other software development site. The results of the multiple-case study should be interpreted as such.

While the objective of the study was to select cases at an organisational level, the practicalities of the cases were narrower than this. As such, the unit of analysis at software division level has multiple embedded units, in line with Yin’s observation. First, the focus of the empirical study was limited to one division of the respective company, although interactions with other company divisions were discussed with participants. Second, data was collected from a subset of the organisation divisions’ range of projects. While the size of projects involved in the study varied, larger projects were comprised of multiple teams spread across multiple countries. For such large projects, not necessarily all project teams were represented by the participants involved. The participants involved in the study were based in the US, Ireland and India. While several phone interviews were conducted with the US and Asia, the majority of interviews were face-to-face at the companies’ respective sites in Ireland. As the study did not have a particular focus on any one part of the organisation, participants were sought from a range of roles in the companies, from site managers to developers. As a summary, while the ideal unit of analysis for this study was a software division, more refined subsets of the unit of analysis were involved during data gathering.

4.4.4 Selection of cases

Wang (2007) argues that when the multiple-case study approach has been chosen, and when the unit of analysis has been defined, two further questions arise in the design of the study: how many cases to include, and how to choose them. The selection of potential cases can be driven by the initial research questions, which later define which data to
collect in those cases (Eisenhardt 1989). As already discussed, the selection of cases was opportunistic as it involved capitalising on pre-existing relationships with industry contacts. It was felt that carrying out a multiple-case study, given the opportunity available to the researcher, would be of value as it would allow for the comparing and contrasting of different cases. For all potential cases, the companies satisfied initial selection criteria such as they all had a strong software development presence in Ireland, and they each actively practiced GSD. All potential companies to be included in the multiple-case study were also within a two-hour travel radius of the university, allowing for relatively easy travel as required.

So as to optimise the findings of a case study, Pettigrew (1989) suggests selecting extreme situations with critical incidence or social drama. While this approach has its merits, it doesn’t, however, encourage the research of ‘normal’ or ‘typical’ cases. This might lead to findings with an exaggerated account of factors involved in the phenomenon of study. Yin (2003b) advises on the selection of cases so as to allow for either literal replication or theoretical replication, by either selecting cases that predict similar results, or selecting cases that predict contrasting results but for a ‘predictable reason’, respectively. In fact, the researcher did not have enough background information of the cases so as to predict the nature of the findings of each case. Rather, it was accepted that cases satisfied the initial selection criteria as already stated. Kirsch (2004), who selected cases based on initial criteria such as similarities between cases including dimensions, took this to be adequate to allow for literal replication.

4.4.5 Data collection

Case studies may combine data collection methods such as archives, interviews, questionnaires, and observations (Huberman and Miles 2002, p. 9). Qualitative researchers typically rely on four methods for gathering information (Marshall and Rossman 1999):

- Participation in the setting
- Direct observation
- In-depth interviewing
- Analysing documents and material culture
Indeed, data collection methods may be combined. Qualitative interviews are one of the most important data gathering tools in qualitative research (Myers and Newman 2007). Typically, a case study researcher uses interviews and documentary materials first and foremost, where participant observation is not being conducted (Myers 1997, Marshall and Rossman 1999, p. 84, Yin 2003b).

This study’s primary qualitative data were obtained through qualitative semi-structured interviews with practitioners involved in GSD. The interviews were used as a tool in order to elicit interviewees’ understandings of various aspects of GSD and its potential benefits.

Kahn and Cannell (1957, p. 149) describe interviewing as “a conversation with a purpose”. The interviewer helps uncover participant’s views but otherwise respects how the participant frames and structures the responses (Marshall and Rossman 1999). This is a fundamental assumption in qualitative research: the participant’s perspective on the phenomenon of interest should unfold as the participant views it, not as the researcher views it. As such, interviews in qualitative studies are a key way of accessing the interpretations of the informants (Walsham 2006). Interviews may take different forms, including the informal conversational interview, the general interview guide approach, and the standardised open-ended interview (Patton 1990, pp. 280-290, Yin 2003b, Myers and Newman 2007). The researcher followed a “focused” (semi-structured) style of interviewing. In such cases, the interviews may still remain open-ended and assume a conversational manner, but the interviewer is more likely to follow a certain set of questions derived from the case study protocol (Yin 2003b). Walsham (1995b) points to the need of striking a balance between passivity of the interviewer and over-direction. Most commonly, case study interviews are open-ended nature, where the researcher (interviewer) asks key respondents for the facts of a matter as well as for the respondents’ opinions about events (Yin 2003b, p. 90). With respect to this point, the researcher was wary of the balance required between posing probing questions of the interviewees to uncover a point of data of interest, and allowing the interviewees enough time to articulate and express their views on the general topic.

The interview data collection method has particular strengths (Marshall and Rossman 1999). Interviews allow for the collection of large amounts of data quickly. When conducting interviews with groups, the process gathers a wide variety of information across a larger number of subjects than if there were fewer participants – a trade-off between
broadth and depth. Interviewing also brings with it some problems and pitfalls (Myers and Newman 2007), as summarised in Table 4-3. In that table, each issue is given a brief summary, then it is explained how that limitation or pitfall was dealt with during the study.

Table 4-3. Issues with conducting qualitative interviews, and a summary of how they were addressed by the researcher. Adapted from Myers and Newman (2007).

<table>
<thead>
<tr>
<th>Issue</th>
<th>Summary</th>
<th>Approach in this study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificiality of interview</td>
<td>Interrogating a stranger to create opinions under time pressure.</td>
<td>Certainly experienced. Researcher was friendly and open to interviewees, and putting them at ease by confirming that data collected was fully confidential.</td>
</tr>
<tr>
<td>Lack of trust</td>
<td>Interviewee may not indulge sensitive data.</td>
<td>Again, researcher confirmed at the beginning of each interview that data was fully confidential and would not be shared with their colleagues.</td>
</tr>
<tr>
<td>Level of entry</td>
<td>Researcher must make contact with person who has reasonable authority.</td>
<td>Contact was initiated with people who had already expressed interest in work of research group. It was important to maintain on-going contact with people of reasonable authority.</td>
</tr>
<tr>
<td>Elite bias</td>
<td>Bias by interviewing “stars” of the organisation.</td>
<td>Interviewees were selected from a wide variety of positions, chosen for their active roles in GSD, thus helping to guard against this bias that may arise if all interviewees were from the same team and level of authority.</td>
</tr>
<tr>
<td>Hawthorne effects</td>
<td>Interviewer may potentially affect the social setting that is under investigation.</td>
<td>It was not an issue of high importance, as researcher did not interfere with day-to-day activities. Rather, allowed for interviewees to reflect on GSD, away from their desks.</td>
</tr>
<tr>
<td>Constructing knowledge</td>
<td>The interviewer must acknowledge that they themselves are creating knowledge in order to understand the phenomenon.</td>
<td>An important factor. Entire research process was a demanding learning process for the researcher. Opinions expressed during the interpretive study by interviewees may not have been fully understood by the researcher because of lack of day-to-day context with the software development process. One counter strategy by the researcher was to clarify with interviewees opinions they expressed whose meaning did not appear to be clear.</td>
</tr>
<tr>
<td>Ambiguity of language</td>
<td>The nature of spoken language will always lead to some ambiguity of what was meant.</td>
<td>Again, interviewees were asked to clarify during interviews. All interviewees agreed that they may be contacted afterwards for clarifications, and several clarifications were provided upon request by e-mail.</td>
</tr>
<tr>
<td>Interviews can go wrong</td>
<td>Potential to unintentionally offend interviewee, for example.</td>
<td>This is a potential hazard in any social context. One telephone interview was cut short, however, due to a lack of understanding between the researcher and the software developer based in India.</td>
</tr>
</tbody>
</table>

Finally, while generally guided by an interview protocol, the semi-structured interviews were reflexive, in that responses to certain questions were allowed to stimulate new awareness and interest in particular issues which could then require additional probing. This strategy is also recommended by Eisenhardt (1989), who labels it “controlled opportunism”. This approach allowed for the interviews to follow questions laid out by the
interview protocol, while being adequately flexible to allow for interviewees discuss additional factors that they felt to be important in the given context.

4.4.6 Interview protocol

The interview protocol was a document prepared by the researcher, containing specific questions to pose, along with a list of themes to be covered during the interview. A section of this interview protocol is displayed as Appendix A. The interview protocol incorporated open-ended questions regarding the benefits of GSD, while being structured in accordance with the framework. This was to give the flexibility to interviewees to express the issues that they deemed to be important to the subject. After the first several interviews, the researcher adjusted the interview protocol to limit the number of fully pre-written questions, favouring instead the list of quick mental pointers for questions to be asked. It was necessary in order to be able to maintain a natural flow to the interview.

Patton (1990) argues that an interview guide is useful for focusing interviews and can also be used as a descriptive framework for analysis. The framework was presented to industry practitioners at a university-hosted workshop on 10th July 2005. The framework was then refined following feedback. The interview protocol for this study was based upon both the benefits and challenges identified by the GSD framework of Chapter 2. The general structure of questioning during interviews was to discuss the intersection between the distances and processes identified in the framework. Participants were asked about both the benefits and challenges identified in the framework. Discussion then moved to understanding the realisation of the benefits, and trying to understand why some of the benefits were not being realised.

4.4.7 Data analysis

Analysis of qualitative data consists of examining, categorising, tabulating, testing or otherwise recombining data in order to attempt to address the research objectives of the study (Yin 2003b). Rather than being driven by an analytical tool or formulas, the analysis of data should be driven by the researcher knowing what to look for within the data. Yin states that the case study method is one of the least structured approaches of dealing with data, although Miles and Huberman (1994) offer useful guidance for researchers. For Yin there are three general strategies for analysing case study data: 1) relying on theoretical
propositions, 2) thinking about rival explanations, and 3) developing a case description. Furthermore, Yin offers five specific analytical methods to apply. One of these strategies is ‘cross-case’ synthesis which is highly suited to multiple-case studies, such as this one.

After taking initial pointers from Yin, the data analysis strategy of this study followed guidance from the sourcebook by Miles and Huberman (1994). They state from the outset that their stance is in agreement with the interpretive point of view that knowledge is a social and historical product, being created by interpretations of interactions between people and with their environment. The fundamental process consists of sifting through written data – that is transcripts of interviews and documentation, along with annotations written by the researcher while in the field. Indeed, some potential data may be lost if it’s not deemed as ‘writable’ and ‘readable’.

Data analysis process

Miles and Huberman state that the analysis process consists of three concurrent activities for getting the most from qualitative data: 1) data reduction, 2) data display and 3) conclusion drawing and verification. Such a strategy is necessary as qualitative data can span hundreds of pages of source data. In fact, these data analysis activities not only happen after the data has been gathered, but rather they evolve along with the data collection process. The researcher must be constantly analysing the data being collected, manipulating, reducing and displaying the data so as to begin to make sense of themes emerging from the data.

Data coding

Miles an Huberman (1994) argue that conceptual frameworks and research questions are the best defence against the overwhelming amount of qualitative data that can be collected in the form of words. This data in its raw form is transcripts from interviews, scribbled field notes, and post-contact summary notes. Codes are tags or labels assigned to varying chunks of the data due to their interpreted meaning (not simply labelling word occurrences). Codes are a way for the researcher to later find ways of accessing and grouping the data. A code can be used to simply describe some data, or can have a deeper meaning, being an inferential code. Miles and Huberman suggest starting with a seed list of codes, as was implemented in this study. As the researcher reads and analyses the interpretations being expressed within the data, it is appropriate to add new codes, and to refine existing ones.
These newer labels will be better grounded empirically than the starting list. Data should undergo a first level of coding even before the next data is collected. In the tradition of grounded theory, this first open level of coding is termed as open coding (Strauss and Corbin 1998). Open coding concerns the initial labelling of data, and also the grouping of these concepts into categories. Strauss (1987), cited by Miles and Huberman, states that the process of coding and recoding is complete when the coding is “saturated”, that is when all events can be easily classified, all the required categories have been created, and sufficient numbers of “regularities” emerge.

Simply labelling the qualitative data usually is not enough; the researcher needs to further abstract the data to infer meanings from it. Pattern coding is the process of further reducing the data, grouping together summaries of the data. It is more an analytical process than simple first-level coding, and has several important benefits for the researcher: 1) it helps reduce the data, 2) it helps the researcher to analyse the data, 3) it helps the development of a cognitive map, and, 4) it helps the multiple-case study researcher by laying the foundations of cross-case analysis by identifying similar emerging themes from the cases.

Axial coding, a term from grounded theory, is perhaps stronger than pattern coding as described by Miles and Huberman. In axial coding, concepts (the ‘spokes’) are grouped around a hub, or axis. Axial coding is a step towards the creation of theoretically rich core categories. The clustering of axial coding can bring together perhaps different coded elements, in effect reassembling coded data (Strauss and Corbin 1998).

Memoing

Along with pattern coding, the researcher should make margin notes where appropriate, and from that should document reflexive remarks on the data. As the data can be descriptive and absorbing, the researcher will be developing their thought process on the phenomenon throughout the coding process. Miles and Huberman suggest formalising this process by creating conceptual memos. A memo is “the theorizing write-up of ideas about codes and their relationships as they strike the analyst while coding” (Glaser 1978). Memoing is the important step in the researcher’s thought process moving from simply reading raw data, to making sense of it and beginning to theorise from it.
Within-case analysis

Eisenhardt (1989) suggests that analysis should first treat each case separately. This is due to the sheer volume of data that the researcher must cope with. Miles and Huberman (1994) suggest ways in which to look at these bounded contexts. Overall, a within-case analysis begins by describing the intricacies of the case, but will later develop to become “basic material for explanations”. Describing a case lies in with the process earlier identified as data reduction, as a description in this context essentially implies the reduction and abstraction of data to be able to understand it. Explanation can be more complex. Bernard (1988), cited by Miles and Huberman, states that explanation is making complex things understandable by using rules to somehow fit together their component parts. To describe a case, Miles and Huberman strongly suggest making use of data displays in order to systematically display summaries of the data from which the researcher (and later, the reader) can draw conclusions. In fact they argue that valid analysis is driven by data displays. Working from a data display, the researcher should develop analytical text based on the display to begin to draw conclusions from the data.

Cross-case analysis

The within-case analysis described above relates to looking at each case as an individual unit. However, the full potential of a multiple-case study is not realised until the researcher draws from the comparisons and contrasts between each case. As with within-case analysis, Miles and Huberman (1994) suggest beginning with building cross-case descriptions, which in turn lead to more explanatory constructs. This study followed what they describe as a case-oriented approach to cross-case analysis, rather than a variable-oriented approach that would be theory-centred from the start. A case-oriented approach is good at finding specific, concrete patterns common to small sets of cases.

There are a couple of benefits that arise from performing cross-case analysis. The first benefit is generalisability (which is further discussed in the next sub-section). It is not appropriate to this study to think of generalisability as a numerical value that can be satisfied by performing a certain number of case studies. Nevertheless, Miles and Huberman still argue that the cross-case analysis can somewhat satisfy human curiosity on whether a single case was a unique once-off occurrence, or whether it lies in with typical examples of the phenomenon. A second benefit is that it allows for deeper understanding and explanation. The analysis performed on each case alone can be leveraged to form
conclusions that are greater than the sum of its parts. That is, a cross-case analysis can be stronger than the sum of independent within-case analyses. Multiple cases can help pin down specific conditions under which a finding will occur. When performing a cross-case analysis, the researcher must find a compromise between maintaining the uniqueness of each case, with being able to generalise from comparisons between them.

Reliability and construct validity, and generalising

Interpretive research has been questioned in relation to dangers of lack of reliability and validity. While reliability and validity are inter-related, each is discussed separately. A potential threat to reliability of a study’s findings can be the researcher himself; Scott (1965) identifies a central problem with the case study method very well: “The sustained researcher who burrows deeper and deeper into a single situation is faced with the danger of emerging so impressed with the complexity and uniqueness of ‘the one dear case’ that he may have difficulty in thinking abstractly about his materials or in attempting to generalize from them”. Walsham (1995b) notes that even if the researcher attempts to remain as an outside observer, he may over time affect the interpretations of the study’s participants. In order to ensure optimal reliability with regard to the research process, the researcher has tried to follow as rigorously as possible the research processes as laid out by others including the planning and design of the study, how the interviews should be conducted, how data should be analysed and reported. Furthermore, the research process was reviewed rigorously during the process as a result of the writing and submission of research papers describing the research involved. Collaborating with more experienced colleagues allowed the researcher to closely learn from their experience with conducting research.

Validity of a research project is focused on whether the actual research in practice matches what it claims to be about. There are several categories of validity; most importantly for interpretive research are construct validity and external validity. Construct validity deals with the extent to which the constructs as operationalised relate to the research phenomenon being studied. Yin (2003b) describes three tactics to deal with construct validity: the use of multiple sources of evidence, the establishment of a chain of evidence, and to have key informants review draft case study report. This study addressed construct validity as follows:

- Within each case, the same set of themes of questions was posed to each of the participants through the interview protocol. The study developed as more
interviews were conducted. While not affecting the anonymity of their colleagues, the researcher introduced questions regarding opinions expressed by previous interviewees. This allowed the study to be informed by multiple sources of evidence regarding particularities of the case.

- A chain of evidence was formed through the on-going recording of field notes during the interviewing process, including notes taken immediately following each interview. The record taking continued during the analysis of data through the researcher’s study notes on paper and codings recorded in the software analysis tool.

- Both case study companies were involved in reviewing the findings of the case studies. This process was used to clarify questions for the researcher and to validate the findings that were formed. The researcher demonstrated the application of the GSD framework at a university-hosted workshop, several companies including Semicon agreed on the appropriateness of the structure of the framework. Initial findings were presented at each company, and their feedback was used to clarify aspects of the findings reported.

External validity is concerned with establishing the domain to which a study’s findings can be generalised (Yin 2003b). Critics typically state that single cases offer a poor basis for generalising. Case studies rely on analytical generalisation, where the investigator is striving to generalise a set of results to some broader theory. Lee and Baskerville (2003) argue for the ultimate futility in attempting to improve the generalisability of a study by introducing further cases. They present an overarching framework that proposes four distinct categories of generalising, only one of which corresponds to the statistical sampling-based generalisation.

Since the information collected for case study research is often specific to the particular situation at a particular point in time, results are not necessarily lead to generalisations in the traditional “statistical” sense (Yin 1994, p. 30). In statistical generalisation, an inference is made about a population (or universe) on the basis of empirical data collected about a sample (Yin 2003b). A fatal flaw in performing case study research is to conceive of statistical generalisation as the method of generalising the results of the case (Yin 2003b). Multiple cases should be considered as multiple experiments or multiple surveys, rather
than being seen as units for sampling. In this respect, case studies can lead to “analytic
generalisation”, in which previously developed theory is used as a template with which to
compare the empirical results of the case study. The result of a case study may therefore be
generalised to theory, rather than generalised to a more universal population. Indeed, the
initial selection of cases for a case study may be theoretical in motivation rather than
statistical (Eisenhardt 1989). By carefully analysing the resultant data of the cases, theory
may be formed from the findings, as described by Eisenhardt (1989).

4.5 Research study implementation

This thesis is based on an interpretive qualitative multiple-case study that involved two
international companies with substantial GSD operations. The two companies are referred
to as ‘Pennysoft’ and ‘Semicon’ in order to protect their anonymity, conforming to non-
disclosure agreements. Research activities involving the companies spanned from 2005 to
2008, from initial heads-up meetings to final feedback meetings of the research findings.
Figure 4-1 below illustrates the stages of research that were carried out in order to
implement this empirical study of the benefits of GSD. The approach to planning the
research study, data collection and data analysis followed the research design as already
described in this chapter, the implementation of which are described in this section.
The following sections describe how this research strategy was implemented in each of the steps above.

4.5.1 Framework refinement

Although this study did not set out to prove a hypothesis, it was important that the study be driven to answer specific research questions. These questions themselves evolved while working on the critical literature review that resulted in the framework of opportunities and challenges in GSD (Chapter 2). The researcher began by studying the broad range of literature related to GSD. During the literature review, it became apparent that specific issues and benefits of GSD mainly dealt with specific characteristics of certain distances
found in GSD. The framework was then structured in a tabular format, with an intersection between each distance and software development process, as identified in the literature. A basic version of the framework with data points was developed in cooperation with the researcher’s colleagues, and was presented at the International Workshop on Distributed Software Development (Ågerfalk et al. 2005).

In June 2005, the researcher presented an overview of the framework of opportunities and challenges to invited industry practitioners. Representatives from four multi-national companies were invited to this university-hosted event, with three of the companies attending, including a representative from Semicon. There was general agreement that the structure of the framework was suitable to studying the phenomenon of GSD – it allowed for a broad enough view of GSD while helping to structure understanding of it. Following the workshop, the researcher refined the initial contents of the framework and used it to further refine the research objective. The process of using the conceptual framework to inform the research questions, and vice versa, is suggested by Miles and Huberman (1994). The focus on the benefits of GSD was seen as a good motivation to further the study, and so the specific research questions were decided upon, while being refined during the design of the case study.

4.5.2 Access negotiation and selection of interviewees

For this study, two cases were chosen as the researcher believed that the exploratory study would benefit from drawing from multiple cases. The choice of companies in the study was opportunistic – it built upon industry relationships that had been initiated by the research group. A total of four multi-national companies were targeted for the study.

After heads-up meetings with management from each of the four companies, only three of the companies agreed to participate in the study given the availability of their resources. One of these three companies initially allowed for interviews to take place. After the first four interviews were conducted there, the head of the software development group did not see the value of an academic study of his organisation, and later declined to continue his organisation’s participation in the study. The researcher decided that there was insufficient data gathered from that case. Therefore, this case unfortunately does not feature in this empirical study. Participation from the two remaining cases continued through the empirical study, upon which the analysis and findings are based.
Both companies requested that non-disclosure agreements be put in place, protecting their anonymity. As the unit of analysis did not focus on a particular sub-section of organisational hierarchy, it was decided to peruse interviews with a range of different people in each company. The reasoning of this decision was to uncover the views, opinions and experiences of people across the spectrum of their organisation, not restricting the study to a specific level within the organisation. Each company provided the researcher with a contact within the organisation, and they were asked to help identify interviewees across the spectrum of factors including level of authority, role, level of experience with GSD, location, and gender. The representatives replied with a suggested list of interviewees and a summary of their roles in the company. The suggested interviewees were deemed appropriate to the study, and the interviews were conducted.

The researcher requested in advance that an audio recording may be recorded at each interview. He explained the importance of having audio description in order to have an accurate recording of the data. Pennysoft agreed to this, but Semicon rejected the request. It was clear that the Semicon was sensitive to corporate confidentiality, despite a non-disclosure agreement being in place. Further, the company requested that the notes from the initial heads-up meeting be reviewed. The researcher complied, and the company made one request, that specific product names be censored. The transcripts from each interview were not reviewed by the company, as they were kept in confidentiality.

4.5.3 Research sites

The two companies, Pennysoft and Semicon, are introduced here. Both companies manage global large-scale software development projects, although neither company can be described primarily as a software company.

Pennysoft

Pennysoft is primarily an investment company. It is a large privately owned US-based company which provides financial services, investment resources and employer services. As of March 2008, the company managed assets worth US$1.5 trillion, and had over 70 million customer accounts (including mutual fund account and brokerage accounts). Pennysoft is headquartered in Boston, USA with a 40,000-strong workforce in 2008. It is in fact made up of two closely-cooperating companies, one serving North America, and one
serving the rest of the world. Those two parts in turn operate a dozen sub-companies, each providing a certain set of services. These sub-companies, or business units, work closely together where appropriate. In practical terms, they act as different groups of the same company.

The Pennysoft Systems Company (a business unit of Pennysoft) opened its European “offshore development centre” in Ireland in the mid 1990s. The Systems Company provides technology solutions, systems and services to the other business units worldwide. For example, they develop Internet-based applications, advanced voice and data communication and other technological infrastructure. The focus of this case study is on the company’s site in the west of Ireland, established there in 2001. The site employs approximately 100 people, and continues to expand with a successful research and development operation there.

Being a financial services company, the software products being developed by Pennysoft involve financial software systems. An example of such a system is the processing and transmission of real-time stock exchange data. Furthermore, the Systems Group creates software systems for internal usage by other business units. An overview of the projects involved in this empirical study, and the interviewees, is provided in the following chapter.

Pennysoft is strongly driven by cost savings. This general policy has led Pennysoft to globalising their software development activities. In the mid 1990s, several sites were established in Ireland, one of which is the case site. Afterwards, the company sought cheaper labour, and set up significant operations in India. There is close cooperation between the US, Ireland and India in the on-going operations involved with their software development activities.

Semicon

Semicon, a NASDAQ-quoted company, is a leading manufacturer of chips and of other computer, networking and communications products. Semicon is a manufacturer of semiconductors with a strong engineering ethos. Its primary goal is to sell its hardware products, and it has substantial software development efforts in order to ‘enable’ its hardware. In 2007, the company had revenue of in excess of US$30 billion and employed in excess of 80,000 people worldwide. Semicon was established in California, and so has roots in the US West Coast approach to life. Semicon’s range of processors are commonly
found in consumer goods. It is a high-tech company, driven by innovation to push its products in the marketplace.

The nature of software developed by Semicon is quite low-level, as it deals with direct interaction with the company’s hardware. Software being developed, for example, is device drivers and software to run networking devices. The company’s case site has expertise in networking and security devices. The site was acquired by Semicon in 2000 and employs 160 people, of which approximately sixty are software engineers. The acquisition was made because Semicon wanted to move into the network processing market, in which this site was specialised. The site is the European research and development centre of Semicon’s largest business group. The group specialises in networking and communications, developing network processors optimised for packet processing.

While software is not the primary focus of the company, the software systems being built and designed can be complex and involved multiple teams distributed globally. Of focus in this study are the operations between the US, Ireland and Malaysia. Similar to Pennysoft, Semicon has established sites in lower-cost economies – for example it has 500 engineers in Gdansk, Poland, and several thousand engineers in Bangalore, India. The specific projects involved in this empirical study are further described in the following chapter.

4.5.4 Data collection

Details have already been provided on the negotiation of access to the two companies involved. Table 4.4 provides a summary of research activities that followed. A total of twenty-five interviews took place across both companies. Of those twenty-five, four Pennysoft interviewees were based outside of Ireland. Beginning in 2005, a heads-up meeting was held at both companies. A representative of Semicon then attended a university-hosted workshop presenting the initial framework. After twenty-two of the interviews had been conducted, both companies attended a second university-hosted workshop where the initial findings of the research study were presented. Following this, the researcher travelled to both companies to present more developed findings, and company representatives provided their feedback. This step allowed for the researcher to ask some further questions to clarify his understanding of the data. Given work on a separate project, the researcher opportunistically requested three further interviews at Pennysoft, which they allowed for.
Table 4-4. Summary of activities for collection of empirical data.

<table>
<thead>
<tr>
<th>Company</th>
<th>Date</th>
<th>Research Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pennysoft</td>
<td>Mar 2005</td>
<td>On-site meeting with Pennysoft management.</td>
</tr>
<tr>
<td>Semicon</td>
<td>Jun 2005</td>
<td>GSD industry workshop hosted at the University, attendees included Semicon. Framework (Chapter 2) was presented to practitioners, and feedback was used to refine categorisation.</td>
</tr>
<tr>
<td>Semicon</td>
<td>Jul 2005</td>
<td>On-site meeting with Semicon management.</td>
</tr>
<tr>
<td>Pennysoft</td>
<td>Jul 2005</td>
<td>Four face-to-face interviews.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project manager (x3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Technical project manager</td>
</tr>
<tr>
<td>Semicon</td>
<td>Aug 2005</td>
<td>Four face-to-face interviews.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Software engineer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Staffing manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Software program manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Technical lead</td>
</tr>
<tr>
<td>Pennysoft</td>
<td>Apr 2006</td>
<td>Eight interviews (three face-to-face, five telephone).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Face-to-face</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Senior systems analyst</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project lead (Indian-based)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Telephone</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Software engineer (Indian-based)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Principal engineer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Development manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Senior software engineer</td>
</tr>
<tr>
<td>Semicon</td>
<td>Jun 2006</td>
<td>Six face-to-face interviews.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Technical lead</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Team lead x2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project &amp; integrations manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Staffing manager</td>
</tr>
<tr>
<td>Both</td>
<td>Jul 2006</td>
<td>University-hosted workshop exploring findings.</td>
</tr>
<tr>
<td>Semicon</td>
<td>Aug 2006</td>
<td>On-site workshop with company management feeding back results.</td>
</tr>
<tr>
<td>Pennysoft</td>
<td>Jul 2007</td>
<td>Telephone conference with manager to feedback results.</td>
</tr>
<tr>
<td>Pennysoft</td>
<td>Feb 2008</td>
<td>On-site meeting with management to discuss preliminary findings, final interviews.</td>
</tr>
<tr>
<td>Pennysoft</td>
<td>Apr 2008</td>
<td>Three interviews (one face-to-face, two telephone).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Face-to-face</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Telephone</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Software engineer (Indian-based)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Project director (US-based)</td>
</tr>
</tbody>
</table>

Interview preparation

In preparation for the interviews, the company representatives were provided with a short summary of the research objective, an overview of the GSD framework and sample questions, and asked to pass them on to the interviewees. These documents were provided
so that interviewees were somewhat aware of the context of the study before entering the interview. This allowed the researcher to begin interviews without much time being spent on explaining why these questions were being asked, and it allowed for the interviewees to briefly reflect on the subject before entering the interview.

Interview structure

A total of eighteen of the interviews were held face-to-face. The remaining seven interviews were held over the telephone, all of which involved Pennysoft. Of those seven telephone interviews, three were to people located outside of Ireland. Four telephone interviews were held within Ireland, due to scheduling constraints between the researcher and the practitioners.

For face-to-face interviews, the researcher and the interviewee sat alone in an office at a desk. For all interviews, the researcher began by introducing himself and briefly describing the objectives of the empirical study. The researcher emphasised that interviews were private, that interview transcripts would be private, and that the participant would remain nameless in the study. For all Pennysoft interviews (both face-to-face and over the telephone), the researcher ensured that the interviewee was happy with the interview audio being recorded. Since Semicon did not allow any audio recordings for corporate security reasons, extensive handwritten notes were taken during interviews, and these were more completely transcribed immediately after each interview session. Such transcripts were not as extensive as transcripts taken directly from audio recordings, but were immediately re-written following each interview, or set of interviews. At the end of each interview, the researcher asked if he may contact the interviewee by e-mail if there were any further follow-up questions. Several interviewees were further contacted by e-mail to clear up any issues after the interviews had taken place.

Interview duration and scheduling

The interviews generally lasted one hour. Several project managers allowed for their interviews to stretch to about 75 minutes. However, one interview lasted approximately 20 minutes. This was a telephone interview with a software developer based at a Pennysoft site in India. It was difficult for the researcher and the interviewee to understand each other, and so the interviewee was not very forthcoming. The audio quality made it difficult to fully comprehend what the other was saying. Indeed, this interview was a good example
for the researcher of how it can be difficult to communicate with someone who is at a remote location, where the communication technology is not of adequate quality, and where both parties come from different national cultures.

The first four interviews at Pennysoft were scheduled to be immediately after each other over a period of four hours. The researcher found that this schedule was too demanding, and that there was not time to reflect upon an interview immediately afterwards. Therefore, subsequent interviews were scheduled so that there would be a minimum gap of one hour between interviews. This allowed the researcher to record thoughts coming immediately from doing the interview, and to further develop written notes that were taken during the interview.

**Interview timeline**

It can be seen given the table of research activities, that data collection for Pennysoft took place in July 2005, April 2006 and April 2008. For Semicon, interviews took place in August 2005 and June 2006. After initial analysis of the 2005 data, the researcher conducted further background research on the research method. A request was made for the next set of interviews, and these then took place in April and June 2006. There was some overlap in interviewees between 2005 and 2006 (two Pennysoft interviewees participated in both, and one Semicon interviewee). This overlap allowed for a certain temporal dimension in the empirical data, allowing the researcher to discuss progress, changes and emerging trends with these interviewees. While this overlap did allow for increased understanding of the phenomenon, the researcher had not designed this study to be a longitudinal one. The research approach was to include additional interviewees where possible rather than re-visiting all former interviewees. Thus, while the majority of interviews took place over two years (2005 and 2006), the study was not necessarily suitable for analysis as a longitudinal study.

The final three interviews at Pennysoft were useful in that they allowed the researcher to further reflect upon preliminary findings with the help of interviewees. By this time, managerial priorities at Semicon had changed, and the researcher failed to secure a final set of interviews there, which may have provided with further insights into the research findings.
During the process of interviewing, it became evident that there were one or two key informants at each company. Such research participants can be the source of leading pieces of empirical evidence because of their willingness to respond to open-ended questioning (Yin 2003b, p. 90). Whether or not the interviewee became a key informant seemed to depend both on their personality and on their hierarchy within the company (these key informants were managerial). At Pennysoft, several interviewees stood out: two project managers acted as key informants due to their openness of expression and drive to speak out. At Semicon, one staffing manager (who also acted as the company contact point) became a key informant for that case, being pro-active in his insights into the researcher's preliminary findings, giving the researcher a further vital level of understanding of the organisation and the work processes of the teams.

Maintaining access

This multiple-case study was made possible through having access to industry practitioners. The process of doing the study involved on-going access to the companies involved, and in particular maintaining contact with the key contact points. As practitioners were under on-going pressures of work commitments, there was a need for the researcher to ‘sell’ the study to them to help justify their involvement. Indeed, two of the four original companies involved did not proceed with their involvement in the case study. Several elements to approaching practitioners’ involvement helped ensure the completion of this study:

- **Identification of key contacts:** after on-site heads up meetings, each company assigned a contact point that would deal with on-going practicalities of the study. This person was responsible for helping to identify interviewees and to schedule interviews. That person’s role and position in the company ultimately guaranteed on-going research access to their company. Of note, once the Semicon contact person left the company in 2007, the researcher was no longer successful in negotiating further interviews.

- **On-going communication and visibility:** as described in the previous section, there were periods of several months between scheduled interviews. These breaks presented a risk to the continuation of the study. The researcher found upon re-opening communication with the contact points that there was a need to further explain the benefit to the study of further interviews. It was suggested by one of the
contact points that a solution to this would have been to provide them with on-going updates on the latest findings in new publications in the academic world. More generally, the risk of access being cut off during the study could have been mitigated through increased on-going communication levels between the researcher and the company contact persons.

- **Providing benefits for practitioners’ participation.** The researcher’s research group held two university-hosted workshops on GSD during this study. The researcher presented elements of the study at both workshops. However, these workshops were primarily promoted to the practitioners as a networking opportunity. The workshops presented a perceived value to the practitioners in allowing them to learn from peers in other similar companies.

### 4.5.5 Data analysis

Data analysis is an on-going process that strongly overlaps with the data collection process. For this study, data analysis indeed began as soon as data collection had commenced. During the interviews, the researcher jotted down initial observations and thoughts based on what the interviewee was expressing. Following each interview, the researcher wrote down thoughts and conclusions coming from that interview. These notes were later compiled as interview reports.

In order to be able to further analyse the interview data, the researcher transcribed the audio of each Pennysoft interview. Generally, one hour of audio took up to five hours to transcribe. For the Semicon interviews, where there was no audio recorded, the researcher immediately went back to the written transcription that was taken during the interview, and expanded on the transcription from memory. Unfortunately, this process resulted in interview transcripts more sparse than those of the audio transcriptions. Overall, the transcription process was beneficial for the researcher, as it allowed him to become immersed in the data, taking note of further observations where appropriate.

Coding of the collected data (including meeting notes with industry contact, interview transcripts, and interview reports) began after the first eight interviews were completed. Coding was executed through the use of a dedicated software tool named NVivo by QSR International. Coding began with a “start list” as described by Miles and Huberman (1994),
that start list being the benefits and challenges of the GSD framework. Passages of text were initially marked according to codes taken from the framework, but additional codes were allowed to be added where deemed appropriate, while the researcher was making sense of the data. As coding of data is a formal representation of analytic thinking (Marshall and Rossman 1999, p. 155), the codes began to evolve from being merely descriptive labels, to understand the interpreted meaning of the text in context. During the coding process, many theoretical questions and observations of the data arose, and these were recorded as memos linked to the text in question. Further memos were recorded for each overall case, where thoughts on emerging themes were kept.

More abstract categories of coding began to emerge as the researcher dealt with the tension between the meanings associated with different coding categories. Analysis involved comparing the incidents covered by these more abstract categories, feeding into the process of axial coding. The coding software tool helped this process by allowing the researcher to consult all texts that had been coded with a specific set of codes. The coding process was intertwined with the conduct of interviews, and certain emerging themes became more obvious when described specifically by key informants. Such themes identified during ongoing interviews led to the refinement of previously coded data, applying codes for the emerging themes to data, where appropriate. As categories became integrated and further data collection did not tend to cause any modification of categories, but rather reinforced already-identified properties, the categories were deemed theoretically saturated.

4.6 Summary

This chapter has laid out the fundamental world viewpoint of this empirical study (an interpretive study of qualitative data). It has described the choice of perusing the collection of qualitative data through a multiple-case study ultimately involving two multi-national companies that practice GSD. The research design decisions of the case study approach have been presented, and the overall approach to data analysis was described. Finally, an overall account of how the empirical study was implemented was described including the data collection through semi-structured interviews and the coding process of the resultant data. In the following two chapters, the output of the coding and analysis phase is presented, first as a within-case analysis including emerging themes from each case,
followed by a cross-case analysis built upon both the seed categories from the GSD framework, and the themes that emerged during the coding process.
Chapter 5. Within-Case Findings and Analysis

5.1 Introduction

This chapter is structured in two parts: one part for each case that was undertaken as part of this study. The cases are treated separately in this chapter, with the within-case findings presented separately. The findings in both parts of the chapter represent the process of within-case analysis, as described by Miles and Huberman (1994, pp. 90-91). The findings and within-case analysis presented in this chapter lead us to the cross-case analysis which is presented in the following chapter.

Yin (2003b, pp. 109-115) argues that the preferential analytical strategy is to rely on the theoretical propositions of the study. To that end, the basis of this study was the framework presented in Chapter 2 in that data collection was structured according to the framework. This chapter presents the empirical findings of each case separately as follows:

1. The organisational context of the case is described including the nature of distribution in their software development teams.

2. The empirical findings of the case are presented according to headings taken from the framework, where appropriate.

3. The case is analysed within its own context to distil understanding about the case findings. This analysis lays the foundation for the cross-case analysis which is described in Chapter 6.

5.2 Case 1: Pennysoft

This section presents the empirical findings of the Pennysoft case study. First, the organisational context of the case study site is presented. Second, the team structure opted for by Pennysoft is described, as it was found that this affects the very nature of their GSD activities. Third, the potential benefits expected and/or realised by Pennysoft are described according to headings found in the framework presented in Chapter 2. Fourth, within-case
analysis begins with a summary of the empirical findings with respect to each benefit. Within-case analysis is offered based on themes which emerge from the empirical findings.

5.2.1 Organisational context

The Pennysoft case study site is part of Pennysoft’s Systems Company. This business group provides services to other internal customers, who in turn serve external customers. A majority of the external customers are based on the east coast of the US, close to the company’s headquarters. To that effect, much of the Irish case study site’s work is dependent on colleagues in the US. Moreover, ultimate project ownership generally rests in the US sites. According to interviewees in Ireland, they themselves have extensive experience in developing software systems in a distributed context: that site was the business group’s second site to be established outside of the US (the first being on the east coast of Ireland). Many of the Irish managers involved in setting up this site are still based at the site allowing for their experience to be leveraged in establishing new teams in India. The case study examined five active distributed software development projects. One of these projects (Payroll Interfaces) is used in Figure 5-1 below to help illustrate the organisational structure of Pennysoft. It shows the project director and analysts residing in the US, with the ‘offshore’ (external to the US) project manager in Ireland, and development sub-teams based both in Ireland and India.
Interviewees at Pennysoft were selected from a variety of projects. Some of the managers within the case study site have experience of several of those projects, and were able to relate their experiences between projects during interviews. Their past experiences also helped improve the researcher’s historical understanding of the evolution of the projects. Here is a summary of the five projects involved in the case study:

- The **Payroll Interfaces project** works for the Payroll Business Group which runs human resources payroll systems for external companies that employ thousands of employees. The business group is based in the US. This project develops interfaces between the company’s internal systems and its customers’ systems. The Irish site
was added to the project when that site was established, and was in turn involved in helping to establish the India part of the team. There are 10 team members in both Ireland and India.

- The **I.T. Governance project** develops workflow applications for a 3rd party workflow engine. The project was established in 2003 involving the US and Ireland, and India joined in 2005. Up until then, software development and testing was performed in Ireland. The business product manager and customer-interfacing analysts are located in the US, with project ownership resting with them. The technical delivery manager is located in Ireland and governs the Irish and Indian teams. There are a total of 20 people in the Irish team, and 15 people in India (following rapid growth).

- The **Development frameworks and technologies project** develops applications for the Enterprise Technology group. This external customer is based in the US. The work involves J2EE development with some ‘green field’ projects, and other application customisation projects. The project had been located in Boston, but was distributed within the US and into Ireland, and then to India as part of cost saving measures. The team is made up of 14 members, including the project director in the US, one tester in the US, a project manager with several developers in Ireland and three developers across two sites in India.

- The **Market data software** engineering team is part of the Market Data Group which was established in 1996. The group develops infrastructure to handle stock market data. The data is purchased from stock exchanges, and sold on to internal customers. The group is based in the US with 80 to 90 people across three locations. Ireland joined the project in 2004. The team develops in-house tools, and works on the integration of 3rd party vendors’ systems in order to provide APIs (application programming interfaces) to the data. The software engineering team is located across two locations: the US with 12 people and Ireland with 8 to 9 people. Quality assurance (Q.A.) activities are performed in both the US and India. The Q.A. team in India has 15 to 20 people.

- The **Market data infrastructure team** is also part of the market data group. The Irish part of the team (established in 2003) works closely with US (established
around 1993) on the design of servers. The US had been the communications hub in the group, but is now evolving to allow for Ireland and India to deal directly with each other. The US site involves 6 people, with another 6 to 7 people in Ireland.

The selection of interviewees came from each of these projects within Pennysoft. All projects involved team members who were based at the case study site, and all projects were global in nature. Table 5-1 gives a complete list of interviewees who were involved in the case study, including which project they belonged to and their current role in that project. All interviewees were based in Ireland unless specified.

**Table 5-1. Pennysoft interviewee codes, projects, and roles.**

<table>
<thead>
<tr>
<th>Project/Interviewee</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payroll Interfaces</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Senior software engineer</td>
</tr>
<tr>
<td>B</td>
<td>Project manager</td>
</tr>
<tr>
<td>C</td>
<td>Project lead (India)</td>
</tr>
<tr>
<td>D</td>
<td>Software engineer (India)</td>
</tr>
<tr>
<td>M</td>
<td>Project manager</td>
</tr>
<tr>
<td>I.T. Governance</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Principal engineer</td>
</tr>
<tr>
<td>F</td>
<td>Development manager</td>
</tr>
<tr>
<td>G</td>
<td>Former development manager (also former project manager under Payroll Interfaces)</td>
</tr>
<tr>
<td>J</td>
<td>Software engineer (India)</td>
</tr>
<tr>
<td>Development frameworks</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Project manager</td>
</tr>
<tr>
<td>I</td>
<td>Project director (US)</td>
</tr>
<tr>
<td>Market Data</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Project manager</td>
</tr>
<tr>
<td>L</td>
<td>Senior systems analyst</td>
</tr>
</tbody>
</table>

5.2.2 Team structure

Figure 5-1 previously illustrated the structure of the Payroll Interfaces project as a sample project structure within Pennysoft. The project was initiated by the HR Business Group. A project director in upper-management resides in the US and is ultimately responsible for
the project. Also in the US are customer-facing business analysts who provide the Irish team with business requirements. The Irish team was established in 2003, and retains active ownership of the project. The ‘Offshore Project Manager’ (Interviewee B) resides at the case study site in Ireland. He has a team lead based in Ireland, who is responsible for four software engineers (including Interviewee A). The project manager also has a project lead located in Bangalore, India (Interviewee C). This part of the team was formed only two years prior to this empirical study. The project lead there is responsible for six software engineers (including Interviewee D).

For Pennysoft, the term ‘team’ was not a strict definition. Interviewees used the term to describe the cross-site organisational structure of a project involving the project director, business analysts, project manager, team leads, developers and testers. The term was also used to define geographical boundaries, such as “the Irish team” and “the Indian team”. However, it is taken that these geographical-based units are in fact sub-units of the overall team structure.

Evident from the illustrated project structure is that the software development team operates across wide geographical distance, and across many time zones between each site. All of the projects involved in the case study are exposed to geographical, temporal, and socio-cultural distance. Teams in Pennysoft are not limited to one site and they typically include members from the US, Ireland and India. Figure 5-2 illustrates a typical structure of a software development team at Pennysoft, with emphasis on the geographical distance within the team.
Figure 5-2 represents a software development team that is spread over three locations. The project director resides in the US along with business analysts, and the project manager in Ireland is responsible for both part of the development team in Ireland and India. This cross-site team structure implies large geographical and temporal distances between each site, in addition to perceived socio-cultural distance. Moreover, it is possible for individual team members to be members of multiple teams at once, potentially leading to a degree of organisational distance. Pennysoft’s organisational culture dictates that a team can also consist of members who in reality belong to different business groups within Pennysoft.

Hierarchy

The emergence of distribution of Pennysoft’s software development activities over time has created a natural hierarchy between the sites. This hierarchical structure affects the type of responsibilities given to each site, and in turn affects the communication structure between them. Interviewee F felt that hierarchy in the company is “overly-complicated” possibly due to the company’s American-oriented culture that promotes a strongly hierarchical structure. In terms of a hierarchy between the different sites in a team, the US sites retain overall ownership of the software projects. Originally, the Irish site was closely managed by the US when first established. Over the next decade, Ireland developed their
expertise to the stage where “it's not a case that [the US are] telling us what to do” (Interviewee A). The Irish and US sites have developed a close working relationship, with several Irish managers having spent years at the US sites. Meanwhile, the Indian site are “not involved in negotiations at present” when deciding which work is completed where (Interviewee B).

India generally maintains a low position in the hierarchical structure due to their relative lack of experience, as evidenced by the fact that their daily tasks are controlled from the Irish site. The Irish site was already well established before the Indian site was established. Generally speaking, team leads in India report back to their superiors in Ireland. As such, the teams’ hierarchical structures span across the sites. The US and Irish management come to a consensus on requirements, and then the Indian site has work handed down to them.

Interviewees in Ireland displayed an implicit perception of superiority over India, referring to “offshoring” work to India. For example, Interviewee B, being based in Ireland, implied that the Irish site were “baby-sitting” the Indian teams. Pointing to being higher in the hierarchy, he stated that Ireland is responsible for India’s actions:

The States are trusting Ireland to make sure that nothing goes wrong... If something goes wrong, Ireland needs to have structures in place so that we responsible for it and that you can react to it. (Interviewee B)

Interviewee E, the principal engineer of his team, described how work is being completed in India on his behalf:

When they’re [India] providing something for me, because really they’re doing some of my work for me so that I can do other work, that’s why we’re outsourcing. For want of a better phrase, they’re doing work in my name, because I’m the one the customer’s dealing with. (Interviewee E)

Overall, the researcher found interviewees from the Irish site accepting responsibility for offshoring work to India. The attitudes of the Irish management towards the Indian site varied between being very satisfied to being frustrated with the work, although at all times displaying a sense of seniority over the Indian teams.
Inter-site communication

While the site hierarchy does affect communication between the sites, open and direct communication is a corporate policy in Pennysoft. From the point of view of the case study site, it communicates on a daily basis with managers and business analysts in the US, while also dictating work to the Indian site. Figure 5-3 below illustrates the hierarchical structure of Pennysoft teams. The figure also shows that there is a constant mixture of both formal and informal communication flowing between the sites.

**Figure 5-3. Typical structure of inter-site communication within one team at Pennysoft.**

![Diagram](image)

Figure 5-3 illustrates the typical communication paths between the sites involved in a single Pennysoft team. The solid lines on the left hand side between the US and Ireland, and Ireland and India represent formal communication such as team meetings. The dashed lines on the right hand side between the sites represent unplanned communication. Unplanned informal communication occurs using email, telephone calls, and instant messaging. Formal communication occurs between the US and Ireland to set high-level goals and milestones for projects. The Irish and Indian teams tend to meet (through teleconferencing) on a weekly basis to discuss the progression of the project. As already mentioned, the flow of unplanned informal communication between sites is quite open, and this type of open communication is promoted by the company. Interviewees reported that such unplanned communication happens on a daily basis between sites.
As Figure 5.3 illustrates, different types of communication flow within the team through the Irish site. This is a result of Ireland’s role within the team – agreeing to the set of requirements with the US sites, and completing the software development activities by closely involving the Indian site. Given the temporal distance positioning of the Irish site, Irish managers can speak directly with the Indian team leads during the morning. After lunch, the site can then coordinate with the US sites. To this end, Ireland’s geographical position is of advantage to the coordination of the team overall:

> When the Americans come in, the Indian people will already be heading home. We wanted to build on that. That comes back to our advantage in Ireland. We can build the bridge between the two. (Interviewee L)

The temporal overlap between the different sites allows the sites to communicate synchronously. In the researcher’s view, such temporal overlap is crucial for the functioning of the cross-site teams, as normal working hours do not allow the US and India to hold synchronous communication. The Irish site facilitates inclusion of both US and Indian resources in the same teams.

However, Irish interviewees expressed some frustration at the effort required to deal with two other sites during the working day. Irish management found it overwhelming having to deal with two other sites, while being responsible for carrying out their own tasks. They felt as though they were “caught in the middle” (Interviewee B). For example, Interviewee H dealt on a daily level with both the US and India:

> We could spend four hours in the afternoon on the phone [with the US]. Would you believe me, I am sick of it. So you’re talking with a woman who’s a bit at the end of her tether. (Interviewee H)

While being the most extreme of the opinions expressed by interviewees on this subject, Interviewee H’s statement shows that it is indeed a difficult task to deal with two separate sites during one day. It also shows that while synchronous communication may be preferred over asynchronous methods, continuous cross-site communication can take its toll on those involved.
5.2.3 Proximity to market and customer

When directly compared to the benefit described in Chapters 2 and 3, Pennysoft does not realise this benefit as described by the framework. Many of Pennysoft’s large business customers are traditionally based in its headquarter city on the US East Coast. Instead of using GSD to set up software development locations closer to the market and customer, Pennysoft actually implements GSD to allow it to move software development activities away from the costlier East Coast cities. As previously described, client-facing business analysts still reside in the US at headquarters. This minimises that upfront effect of GSD on customers:

_The customers are not affected... The customers are not thinking that the development is happening in the US, Ireland, or India._ (Interviewee D)

Further benefits afforded by distributing Pennysoft’s software development activities away from the customers’ locations are discussed below.

**Conclusion:** The proposed benefit is not being realised as Pennysoft is not reaching closer to its customers through GSD. However, a variant of the benefit is being realised, as GSD allows Pennysoft to seek out cheaper labour markets while maintaining their presence close to the market and customers in the US. It allows them to reduce production costs while continuing to charge their customers fees relative to the US market.

5.2.4 Improved focus on documentation and improved process definition

_Both potential benefits of improved focus on documentation, and improved process definition were treated similarly by interviewees, and are described here as such._

When asked whether they thought GSD was beneficial by improving focus on process definition, interviewees generally concluded that they did not perceive this to be a benefit. Managers saw the need for formalising their communication structures, and this includes having “very tight” documentation (Interviewee G). Interviewee G found that well-defined documentation was needed to overcome the obstacle of temporal distance which limited a developer’s ability to “thrash it out” with an analyst directly. Interviewee H found that the lengthy project documentation, however, did not help:
Interviewee D noted that having to maintain cross-site documentation directly increases the need to coordinate with the other site, which is rendered difficult with a high degree of temporal distance. This comment points to the broader issue of the extra effort required for coordinating project elements with a high degree of temporal distance. This distance means that remote colleagues are not always (or ever) available during one’s office hours.

The Irish managers in Pennysoft found that processes must be very well defined when dealing with other sites. Several interviewees, including interviewee C who was referring to his own work, noted that Pennysoft had had process issues in the past with a lack of escalation of issues from the Indian site. The Indian developers had been hesitant to acknowledge that a delivery would be delayed or that they were having problems completing a task. According to Interviewee B, clear processes were therefore put in place by the Irish site:

One of the bonuses that I saw when I came into the team [in Ireland] was that there were very clear processes. [Paraphrasing communication with the Indian site:] “This is the way we want you to do it, and you should not digress from this.” There were very clear communications and checkpoints to see if something needs to be escalated in a certain way. (Interviewee B)

Interviewee C, the project lead of Payroll Interfaces in India, stated that process definition was also helped by the co-ownership of the Irish and Indian sites of a “development process document” to which they agreed to adhere.

Processes were “instilled” in the new Indian teams by Irish management travelling to India to train in their new counterparts (Interviewee B). This indicates that process definition was deemed an important element in establishing a cross-site team. The same interviewee noted that Ireland had processes imposed on it by the US when that site was growing very rapidly earlier in the decade: “we grew so quickly that our quality went down, and we took a big hit”. To counteract this drop in quality, “we then had to put in very very strong processes and checkpoints in place”. For example, all team members had to complete code reviews, and it was their responsibility to reject requirements changes from customers. This shows that it was deemed necessary to impose better defined processes during the high-growth period to
counter-act quality issues during a high growth period. The Irish managers, who were involved with establishing the Irish site a decade previous, learned that such process definition was an important element in expanding their GSD activities further. Interviewee A of Payroll Interfaces attributed an increased sense of teamness between Ireland and India rather than the US to the standardisation of their work practices.

Interviewee I of the Development Frameworks project agreed with neither proposed benefit, when comparing the outcome to co-located development:

*I think standardisation is a prerequisite for success ... This is an area where I think my team is struggling. I think we don't all work the same way, and as a result it's making it very painful. It's forcing us to do it. But on the other hand..., if we could be effective without standardisation, would we really need it? And to some extent, I would say yes. But I would say in order to be effective in the model we are in, we need more standardisation than we would if everybody were in the same location. (Interviewee I)*

Interviewee I was expressing doubt over this proposed benefit, in that documentation and processes are formed in a global context out of necessity, and that co-located projects can survive without as much emphasis on these elements.

Interviewee A, a senior software engineer, noted that while the team at the Irish site have been able to put in place standardised processes with India, the US have refused to adopt new processes proposed to them by Ireland. The aim of the proposed processes was to ultimately improve the quality of code, in the view of this interviewee. The interviewee felt that the standardisation of work practices between Ireland and India strengthened the sense of teamness between them, and that it would similarly have helped to improve relations with the US site if they had been adopted. This points to the opinion that standardised processes may help bind cross-site teams. If sub-teams in separate locations were to work to differing sets of processes, the perceived organisational distance may be greater.

*Conclusion:* It was found that interviewees see the need to focus upon carefully defining processes and documentation due to the nature of GSD, but are questionable in their ultimate positive contribution to the development process.
5.2.5 Record kept of communications

This proposed benefit suggests that a positive aspect of relying on asynchronous communication is that a record is kept of the communication.

Interviewee C, being based in India, stated that a phone call is a faster method of receiving a response from a remote colleague compared to e-mail. However, e-mail is also beneficial in that you have time to frame your thoughts accurately while composing the message, and the interviewee agreed that it is useful to have a record of the communication.

Interviewee F, on the other hand, found that instant messaging can support informal communication precisely because a record is not being kept of the conversation:

_A lot of the informal exchange among team members is definitely using IM, even within [our site]. I mean, people will get up and talk to each other, but, say, I was on a conference call with a person in the US in Boston, and someone in India, and I wanted to say something to someone in Dallas, I can send them an IM to say “what do you make of that?” And they’ll respond. You can have an informal exchange while there is something else going on in the background which is one of the powerful things of it. And there’s no record, which is also a good thing. It wouldn’t be informal if there was a record. Emails are very different._ (Interviewee F)

In that situation, instant messaging added value by allowing developers to converse while being off the record.

**Conclusion:** While Interviewee C agreed with this proposed benefit, the researcher did not find resounding support for the significance of this benefit. Moreover, this chapter finds that team members at different sites shift their working habits with the goal of increasing the possibility for synchronous communication. In the researcher’s view, this tendency found at Pennysoft shows a preference for synchronous communication despite potential benefits attributed to asynchronous communication. We must conclude that while asynchronous communication does have certain benefits, they do not weigh up against the effectiveness of synchronous communication and therefore this benefit is not being realised.
5.2.6 Access to large skilled labour pool

Pennysoft has been able to use GSD in order to access employment markets outside of the US, as evidenced by the establishment of sites in Ireland and India. According to several interviewees, the Irish site was established due to several favourable factors, including the availability of a skilled labour pool, along with relatively cheaper salaries and a governmental policy of low corporate tax. Interviewees based in Ireland expressed the opinion that the Irish site has matured over its first decade, resulting in a maturation of its capabilities through its hands-on dealings with distributed software development, and that this experience is now of value to the company. Promoting this level of experience is seen by the Irish site as a way to justify their existence:

_We’re ten years already in Ireland. We’re very process mature. Ireland understands business better [than India]. It takes time to get to that position. That’s the benefit we have, combined with technology skills._ (Interviewee F)

When asked whether access to a large skilled offshore labour pool was of benefit to the company, interviewees generally expressed that such a benefit was more of a facilitator of the benefits of reduced development costs. As such, Pennysoft is not actively seeking out labour overseas due to a shortage in the home market, but rather, it allows them to hire cheaper staff.

Interviewee D, a software engineer based in India, noted that more home-grown talent is now generally available in India than before. Many Indians had in the past travelled to the US to find employment there. However, the trend has now reversed, with skilled workers returning to India to benefit from the increase in the standard of living there.

It was recognised by interviewees that the benefit of access to a large skilled labour pool is realised incrementally, in that it takes several years for a site to become mature and efficient in software development. The lack of current business experience in India has caused some concern for the Irish site which is ultimately responsible for them:

_I haven’t seen problems with the work we have given India. Most of the issues have come down to experience levels, and support… they do not necessarily have the business experience for all those projects. That can be a problem._ (Interviewee E)
Where a more experienced developer can make decisions independently, one who is less experienced can tend to wait for feedback, which can introduce day-long delays (Interviewee H). Irish-based Interviewees generally agreed that the Indian teams were gradually gaining experience, and therefore reducing the extent of the initial problem. Also, Interviewee F noted that there is a tendency to underestimate the ability of India because of a lack of visibility of the work that they are completing. Similarly, Interviewees B and H felt that the skills and expertise on offer by Ireland were sometimes overlooked by the US because of the company’s priority to offshore work. Overall, there is more pressure to leverage cheaper resources than to target the sites with the most experienced labour pool:

I think that we're being very successful. But in the long term, the dollar rules. And also policy rules. I think that a lot of managers would like to put work in Ireland, but they've been given a mandate to put 40%, say, of their work offshore, which means India. (Interviewee H)

Interviewee H seems to be indicating her belief that the US underestimates the effectiveness of the skills found at the Irish site compared to lower-waged employees in India. The Market Data team found that software development “didn’t work in India” due to a lacking skill set found there. The work in question was pulled out of India and moved back to Ireland (Interviewee K). Interviewees H and I of the Development Frameworks project were the most critical of their experience with searching out skilled labour in India. Their team had attempted to hire a .NET developer with a minimum of four years’ of experience, but did not find one during a thirteen month-long search. They felt that a person with this experience was needed so that they could work quite independently, a requirement for developers in the challenging context of GSD. Interviewee I recalled rejecting two thirds of the initial applicants. However, they did not manage to secure interviews with the remaining job applicants: “we’d sit on the phone and wait, and nobody would even come” (Interviewee I). This effort spanned 2005 and 2006, at which time there was a high demand for such candidates.

Interviewee H expressed dissatisfaction with the Java developers that her Development Frameworks team employs in India. The interviewee complained of having to dedicate 50 per cent of an associate engineer’s time in Ireland to “hand-hold” a more senior engineer employed in India. The Irish team was involved in interviewing candidates for the Indian site, but having a telephone interview did not allow for them to correctly assess for “ownership of work and that kind of all-round competence”. This view in itself indicates that
physical distance between those involved may make it harder to assess the others’ competencies.

They [Indians] don’t have the skills or experiences they’re saying they have. Because it seems like the Indian companies don’t understand what it means to have skills. They think that anybody can come in [and] just do the job. (Interviewee H)

While the previous quote may be excessively generalising, it does display a sense of frustration felt at the Irish site in relation to the skills expected by them in the Indian teams. The interviewee continued to compare her developers in Ireland to her developers in India:

[Our developers in Ireland can say] “look we forgot this”, as opposed to a developer [in India] who barely does what he’s told and will have some mistakes in it. There’s a huge difference in the value to [the company]. Like, one of these [developers in Ireland] is worth more than two of these [developers in India], for sure. But nobody ever models that cost. (Interviewee H)

The interviewee suggested that part of the problem may be the Indian culture where the resources expected to receive daily instructions as to what tasks they would be expected to complete. This in turn increases the effort for Interviewee H, as the Indian resources must be micro-managed from the Irish site.

Conclusion: The potential benefit of access to large skilled labour pool is certainly not a driving factor of GSD for Pennysoft, but rather a facilitator to reduce costs. Meanwhile, the Irish managers complain that the skills in India are lacking, and don’t match expectations according to their assigned level in the organisation.

5.2.7 Access to cost-effective labour pool

Every interviewee shared the opinion that Pennysoft was distributing their software development to ultimately reduce costs. Several interviewees pointed out that these costs include cost-effective labour, but also cheaper real estate and infrastructure. There is a general policy in the company to ‘offshore’ work: “This is aimed at managing the cost of the overall program. We’re not adding heads in the US or Ireland” (Interviewee F). Indeed, the Irish site was founded primarily as a cost-saving measure by the company (meanwhile, the site’s
cost has risen). When asked about Ireland’s continuing role in face of other cheaper sites, Interviewee K said that is was in Ireland’s interest to leverage the lower-cost services of the Indian sites, as this adds value to Ireland’s proposition. The interviewee is expressing a generally positive view, which could otherwise be expressed as fear of cheaper sites.

Pennysoft’s costing model is based on the price-per-hour charged by each site. This encompasses all costs, including salaries, real estate, and infrastructure costs for each individual site. Decisions on how to distribute projects are taken based on these figures, which are presented in Table 5-2.

Table 5-2. Approximate rate per hour, as charged by each site to internal customers.

<table>
<thead>
<tr>
<th>Location</th>
<th>Rate per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>US$110</td>
</tr>
<tr>
<td>Ireland</td>
<td>US$80</td>
</tr>
<tr>
<td>India</td>
<td>US$35</td>
</tr>
</tbody>
</table>

As such, Pennysoft’s operations are three times less costly in India compared to the US, and the company’s GSD policy is heavily influenced by these figures. According to several interviewees, teams have been mandated to offshore 40 percent of their software development activity. As a side note with respect to sourcing cheaper labour abroad, there was not convergence on what “offshore” referred to.

*Also, there was keenness on the part of the US to send work offshore, and that “offshore” in our opinion means India, China, Vietnam. (Interviewee B)*

*You are supposed to have a certain percentage of your team where you aim for offshore. And Ireland doesn’t really qualify for offshore. We’re sort of near shore. It’s very…, it’s difficult to figure. And I’ve asked this before at these type of planning presentations, is Ireland offshore or not? You can’t quite tell. (Interviewee H)*

When asked about this benefit of access to a cost-effective labour pool, some Irish-based interviewees countered the proposed benefit with the issues that they were experiencing with distributed software development. It should be noted that Indian-based interviewees did not make this complaint. Interviewee G felt that it is beneficial to make use of offshore cost-effective labour only if the software process itself is already quite efficient. Otherwise, the interviewee said that coordinating across three time zones can “only make it worse” for
the teams. Interviewee H felt that the cost figures taken into account by Pennysoft’s accountants were not realistic, as they do not take into account the overhead of managing resources across distance:

> It’s what I’d call bean counting. It’s looking at stuff in isolation. Because that’s what accountants do… It’s as if the people who decide all this stuff don’t understand software development very well…. You get a lot more out of having 3 people in the one room, they will develop something, the ideas will come up, the things they’ll find, the quality of your solution will be way better than if you put them at three separate sites. And I think that’s what’s not being factored in. (Interviewee H)

Interviewee I offered that the problem with Pennysoft’s costing model was that software output is difficult to measure, while costs are easier to measure. The interviewee felt that while employing resources in India did allow for the company to pay relatively lower salaries, it is very difficult to measure whether or not the quality of the output remains static regardless of this distribution. He explained this through a metaphor:

> I would liken software development more to writing a book. So if you imagine the book, that would come out if you wrote a chapter, and somebody in India wrote a chapter, and I wrote a chapter… You know, I think people think of software as a manufacturing process… But it’s also got the artistic side too, where as I said it’s more like writing a book, where if the author went home at midnight and didn’t finish the chapter, there’s a real continuity problem if another author came in and tried to finish that. (Interviewee I)

Interestingly, it was noted that cost figures are constantly evolving. Ireland had been viewed as “low-cost” in the 1990s, “but now we’re based on quality” (Interviewee A). Interviewees also reported of rising costs in India, and suggested that China may be the next target for cost savings in software development. Interviewee D noted that Pennysoft had recently opened a site in New Mexico as opposed to expanding their operations further in India, possibly pointing towards the problems of distributing software development to India.

**Conclusion:** This potential benefit is certainly the driving factor for Pennysoft’s GSD activities. Headline figures for hourly rates are US$110 for the US sites, US$80 in Ireland and US$35 in India. Some teams have been mandated to send 40 percent of work offshore in order to reduce costs. However, several interviewees argued that this estimation was not
necessarily resulting in cheaper software due to the effort required in managing the offshore resources.

5.2.8 Cross-site modularisation of work

Pennysoft does not split development activities according to independent sub-components of the software systems being developed. Rather the opposite is practiced, where the Irish and Indian sites work together on a continuous basis to develop software. The systems being developed are not broken into modules to allow for each module to be developed in isolation, as proposed in the literature. Work is naturally assigned to the site that has the best experience to carry out certain responsibilities of the project work. The Indian site is not directly involved in making such assignment decisions. For example, the Market Data group makes use of experienced developers in Ireland, while the less experience Indian team is used for quality assurance tasks.

A down-side to Pennysoft’s approach to team structure is that software development involves daily communication between sites, across geographical and temporal distance. Managers are even responsible for the daily tasks of developers at remote sites. Meanwhile, the interviewees felt that the nature of the work at hand does not lend itself well to modularisation in advance, as it requires close coordination between team members. It was expressed that their “interaction-intensive” work involving a mixture of analysis and technical work cannot be separated in large segments of work responsibility:

There’s not a piece where you can generally go off and say, [although] there’s some smaller pieces alright, but nothing substantial that could say “Go off and do that for 3 months”.
(Interviewee H)

Interviewee F felt that if work could be packaged into coherent pieces and responsibility for them given to India, it would improve India’s morale as it would allow them to feel more independent. This points to an issue with the high degree of site interdependence inherent in the teams’ work. Interviewee B believed that the Indian site is not happy to be dependent on the Irish site, as they are constantly pushing to win better quality projects without Ireland’s involvement.

The Market Data engineering team had been developing software across the US, Ireland and India, but it was felt that this was overly-dispersed. The work, according to Interviewee
K, could not be easily divided and therefore “the development didn’t work in India”. Interestingly, this resulted in the team withdrawing their development work from India, despite it being a relatively lower-cost location.

> A project needs to be big enough to modularise into chunks. With the application integration work, we tried to distribute every task. But some things were too small to distribute. There was no efficiency. (Interviewee K)

The quote above shows that teams in Pennysoft at least consider cross-site modularisation of work to be potentially beneficial to their work. Interviewee H further commented on the effect of project size on the ability to modularise work. The interviewee felt that larger teams (greater than their 14-person team) would allow for more work to be separated more efficiently. In addition, the interviewee felt that the smaller team size can leave team members feel isolated:

> I think that would definitely be better if they’ve got a larger team, you definitely need a larger team. I think smaller teams don’t work. I would say, I feel sorry for the two teams in India that I have, because they’re on their own there, there’s just two of them. You feel more part of a team if you’re with it [at the same location as others]. It’s hard. (Interviewee H)

The previous quote points to the opinion that geographical distance may affect the sense of teamness, at least in the situation where one of the sites does not possess a critical mass of team members. Pennysoft’s lack of ability to modularise their work tasks seems to have lead to certain overheads leading to much added effort, and this is analysed in the section 5.2.16 Emerging theme: Team structure adding to overheads. Interviewee K felt that while the nature of the work means close collaboration across distances, future developments may allow the Indian site to become more independent as they gain business experience. From the researcher’s perspective, such progress may be to the detriment of the sense of ‘teamness’ between sites.

**Conclusion:*** The benefit is not being realised, although certain teams have attempted it. Instead, tasks are not easily separated between sites, with an on-going level of higher cross-site dependence. This forces the teams to communication across global distances.
5.2.9 Time zone effectiveness

The Pennysoft teams span three continents, implying a large temporal distance between team members. The East Coast sites in the US are five hours behind Ireland. There is an 8.5 to 9.5 hour time difference between the East Coast and the Indian site (depending on Daylight Saving Time in the US), allowing for zero hours’ overlap between the US and India. Ireland has a temporal overlap with both India (3.5 hours) and the US (4 hours). These overlapping hours vary according to the time of year. Figure 5-4 illustrates the approximate overlapping working hours of the different sites due to the temporal distance between them.

Figure 5-4. Overlapping working hours between India, Ireland and the US, with forces acting on standard working hours.

Several interviewees agreed that Pennysoft benefits from leveraging the temporal distance between the sites in certain cases. However, a majority of interviewees did not agree that Pennysoft realise this potential benefit. Interviewee C noted that time zone effectiveness can work out well as a benefit if the work is planned for correctly. Interviewee K in the Market Data project described how that team was trying to maximise the number of hours software development takes place during the day. It was only possible to realise with work tasks that could be completed within a working day by one engineer. Such work included configuration of third-party applications, performance testing, and testing whether an application is suited to their systems – therefore these tasks are not purely the coding of software. The interviewee noted that it also takes time to build up such a rhythm because of the forward planning required. Interviewee M stated that her team was implementing the
follow-the-sun model for defect resolution where India hands off work to Ireland and they in turn hand off to the US. The interviewee noted that follow-the-sun is “really tough”, especially for software development work. She suggested that it is only possible to implement when experienced team members are available, and where each site has enough expertise to perform hand offs efficiently.

Interviewees countered this suggested benefit with various issues related to working across temporal distance. Interviewee H complained that her team could not leverage this benefit as the team members in India would not be able to work independently without requiring direct contact with other sites for support. Interviewee G said that working with colleagues across temporal distance can cause delays of up to 1.5 days because of temporal distance between them.

Many of the interviewees commented a negative effect of coordinating across time zones, that is the shifting of working hours in order to maximise temporal overlap with other sites. Figure 5-4 illustrates this tension through the dashed arrows: there is pressure on India to work later with Ireland, on Ireland to work later with the US, and on the US to work earlier for Ireland. Interviewees E and H had experienced issues with US-based colleagues who did not provide the flexibility required in such a working environment as they did not have previous experience with GSD.

However, the company culture promotes maximising the number of overlapping work hours during the day (at the cost of personal time). This was evidenced by many of the managers stating that they and their team members regularly work past contractual working hours.

Despite some problems with the US adapting to a globalised work environment, Irish interviewees did agree that the US sites do try to begin work earlier in their day to increase their temporal overlap with Ireland. The Irish interviewees reported working often until 8pm or 9pm so as to maximise the communication window with the US. Interviewee G noted that the Irish working day is “not that far off” the US working day because of the shifted working hours at both locations. Interviewee D said that their team in India operates on staggered working hours: one part of the team works during normal working hours, and a second part works during the evening so as to have a temporal overlap with the US. When asked whether the developers in India accept such staggered working hours, Interviewee D replied: “See, we are prepared for that. The people know that. Whenever we are working
for multi-national companies. It is the work culture, kind of.” Interviewees noted that meetings involving team members from all three continents are difficult to set up because of the temporal distance:

And the India piece is just a mess. 7am here (in the US) is noon in Ireland, and 5.30pm in India. It’s the only time that kind of works. But it’s either that or people are missing dinner or coming into work before breakfast to actually have a meeting. (Interviewee I)

Finally, Ireland’s geographical position with overlapping working hours with both the US and India allows the site to act as a go-between. This in effect seems to enable the company to offshore work to India more efficiently, as there are no standard overlapping working hours between the US and India. All three Pennysoft interviewees based in India agreed that having Ireland as part of the team helped the team to work more efficiently, compared to having to work directly with the US. Interviewees K and L suggested that this offered a benefit specifically for the Irish site, as it is a selling point for them. However, Irish management expressed quite openly their difficulty with dealing with both India and the US during their working hours. Interviewee B, an Irish manager, expressed dissatisfaction with managers being asked to work out of hours, and questioned how sustainable such a practice would be. Interviewee G complained of having lack of free time, having to deal with India during the morning and with the US during the afternoon. Interviewee C in India also noted that they must “stretch” their hours to deal with Ireland, and even the US when required. This choice of phrase does indicate the effort required by the Indian team members to improve their temporal overlap with remote colleagues.

**Conclusion:** Two interviewees agreed that time zone effectiveness is somewhat realised in their teams. However, it only applies to tasks of small granularity such as testing. It was stated that India is not independent enough to allow this benefit to be realised for more complex development tasks. Moreover, there is pressure on all sites to shift their working hours to maximise their overlapping work hours. In effect, this is reducing temporal distance because of the challenges it introduces, rather than leveraging temporal distance to maximise the number of working hours during the 24-hour day.
5.2.10 Time zone efficiency

The benefit of time zone efficiency was only noted by Interviewee C, a project lead for Payroll Interfaces who is based in India. The interviewee noted that in the context of coordinating with the US, the Indian site can “concentrate on the actual work” during their morning, and then allocate the last two hours of their day to coordinate with Ireland and the US, thus experiencing time zone efficiency during the morning hours. The implication was that while meetings are a necessary part of the GSD processes, temporal distance allowed for the confinement of meetings to a specific section of the day. As noted in the previous section, the Irish site cannot realise this proposed benefit, as they coordinate with the Indian and US sites throughout their day.

*Conclusion:* This benefit is realised by the Indian site during their mornings. It is not realised at the Irish site. Perhaps the US also sees this as a benefit during their afternoon, but insufficient data was collected to establish this.

5.2.11 Innovation and shared best practice

Interviewees did not mention this benefit at any point without being prompted to do so. There were mixed reactions to this proposed benefit. For example, Interviewee K, the Market Data project manager, felt that having a mixture of people from different backgrounds “will always help, [having] different views by different people”. Interviewee D, based in India, expressed a similar sentiment. Interviewee C, also based in India, felt that policies promoting sharing of practices came from the top down, coming from the US and encouraging the sites to share their best practices. Irish-based Interviewee H of the Development Frameworks team stated that they do not realise this proposed benefit and felt that India’s attempt to share their best practices was not worthwhile:

*Their manager would think there is [a flow of best practices] and be keeps on sending me stuff about how to work effectively with teams abroad. And I’m like [frustrated sigh], you know?*

*(Interviewee H)*

It is evident from this quote that there is a lack of respect at the Irish site for the intentions of the Indian teams. Interviewee I, the US-based project director of Development Frameworks, felt that the propagation of mature processes would occur in the company
regardless of the nature of distribution of their teams, and therefore disregarded it as a benefit of GSD.

Conclusion: There are weak signs of this proposed benefit being realised. Interviewees in Ireland and the US were sceptical of its merits. We may perhaps conclude from this that innovation and shared best practices could not naturally flow where there is a lack of respect for the best practices being offered.

5.2.12 Allocation of roles and team structure

Interviewees explained that Pennysoft’s approach to team structure allows for a mixture of people to be included in one team wherever they may be located. Resources can also be moved between projects within the same business group as they become available, maximising the company’s usage of its human resources.

[Where we send the work will] depend on where the expertise is, on where the spare resources are.
(Interviewee L)

This organisational flexibility aims to minimise downtime of isolated resources, and involves assigning engineers to multiple projects in parallel where necessary. However, Interviewee E, the principal engineer for the IT Governance project, felt that the Indian site was not up-front about staffing, thus limiting the potential for benefit:

[India] have to manage all their tasks that they have to do, it’s a waste of a resource to assign them to just one project, as there is inevitably going to be downtime, and probably they should be working on something else at the same time. I believe that they are trying to present to us that this person is working on only one thing, but there probably is other work going on. That’s my perception. (Interviewee E)

The spirit of this potential benefit, as defined in Chapters 2 and 3 is that work can be allocated across different sites depending on resource availability. However, if Interviewee E’s perception is true, then this benefit is being realised more on a localised benefit within the Indian site, rather than being potentially realised in a more global context.

Interviewee H felt that the benefit of allocation of team roles across different sites is perhaps difficult to achieve, as each time a person is moved between projects, they must spend time to familiarise themselves with the project:
There is a senior technical resource who works for [us] who is based [in India]. Management [in the US] are saying, “let’s use this guy”. I say, that’s fine, in theory. But to me, whoever pulls that, thinks of people as little widgets you can move around. [But the person] is no good unless he knows what’s going on in the project… It’s from a cost model as if people were interchangeable, and not taking into account the cost, the overhead associated with distributing people, and with communications and in management. (Interviewee H)

The interviewee continued to state that the allocation of resources in India was unavoidable due to company policies, but that she would “definitely take two people in Ireland instead of three in India” due to the difficulties involved with the distributed work environment and their skills level. Therefore, while the organisational structures in Pennysoft are open to easy re-allocation of resources, there may be a significant friction cost associated with it.

**Conclusion:** The software team structure in Pennysoft is flexible in order to make use of available resources, regardless of where they are located. This is a strength of their approach, as it maximises the use of labour when it becomes available. The benefit on a global level may be limited by lack of visibility of resource availability, although it can be decided locally to assign a developer to more than one project at any time. Interviewee H points to the cost of each re-assignment, as it is costly to bring them up to speed on a project. Moreover, this flexible team structure exposes the team to a high degree of geographical and temporal distance.

5.2.13 Increased autonomy

This potential benefit of increased autonomy was not regarded to be relevant in the context of Pennysoft. Quite the opposite was evident, in fact, with the Indian site struggling to become more independent. Ireland and India have a joint development document to agree upon standardised work practices. This seemingly takes away from the proposed benefit, as these sites do not have process independence from each other. Also, according to Irish management, the Indian site is lacking essential business expertise which would allow it to become increasingly autonomous, although this could change in the future as they build their expertise.

*Conclusion:* Potential benefit is not being realised. The tensions arising from the lack of site independence is discussed in the following sections.
5.2.14 Within-case analysis: Evidence and drawbacks of the benefits

Table 5-3 summarises to what extent Pennysoft realises the potential benefits of GSD according to the findings of the case study, and how they are affected by the distances of GSD.
Table 5-3. The realisation of benefits of GSD at Pennysoft.

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Evidence</th>
<th>Challenges</th>
<th>Realisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proximity to market and customer</td>
<td>• Keeping some development activities on the US East Coast.</td>
<td>• Stretches development activities across multiple continents.</td>
<td>• No, yet some proximity maintained for market and customers.</td>
</tr>
<tr>
<td>Improved focus on documentation &amp; standardised processes</td>
<td>• Necessary in distributed teams. • Shared processes help promote sense of teamness.</td>
<td>• US not implementing processes. • Documentation not adequate for informing new team members.</td>
<td>• No. Neither proposition ultimately accepted as beneficial.</td>
</tr>
<tr>
<td>Reliance on asynchronous communication</td>
<td>• E-mail does help the sender to distil their thoughts.</td>
<td>• Synchronous communication is a necessity with the teams’ current structures.</td>
<td>• No, not ultimately accepted as beneficial.</td>
</tr>
<tr>
<td>Access to large skilled labour pool</td>
<td>• Irish site established in 1990s. • Larger Indian site established in 2000s.</td>
<td>• Not driven by lack of skills at home, but rather a facilitator for access to cheaper labour. • Ramp-up of several years for new sites. • Lack of business experience affects ability to work in distributed team. • Large labour pool, but under high level of demand.</td>
<td>• Yes. Secondary benefit to seeking out cheaper labour costs.</td>
</tr>
<tr>
<td>Access to cost-effective labour pool</td>
<td>• Primary motivating factor for GSD. • Policy to have 40% of labour offshore. • India bills three times less per hour than US. • US at US$110, Ireland at $80, India at $35 per hour.</td>
<td>• Drive for cost reduction results in distributed teams, leading to coordination overheads. • Overhead costs not taken into account, and would be difficult to model.</td>
<td>• Yes, but with considerable overheads.</td>
</tr>
<tr>
<td>Allocation of roles and team structure</td>
<td>• Resources can be moved between teams regardless of location, enabled by distributed team structure.</td>
<td>• Ramp-up time required for each engineer’s re-allocation. • Documentation is not a sufficient aide for familiarisation.</td>
<td>• Yes, thanks to flexible team structure.</td>
</tr>
<tr>
<td>Cross-site modularisation of work</td>
<td>• Recognised as possibly advantageous.</td>
<td>• Not being enforced as an ideal to strive towards. • Nature of work with relatively small team size seen as barrier to benefit. • High degree of cross-site dependence evident.</td>
<td>• No.</td>
</tr>
<tr>
<td>Time zone effectiveness</td>
<td>• Being realised for work such as configuration, performance testing and defect resolution.</td>
<td>• Indian teams not adequately independent. • Shifting working hours to maximise temporal overlap.</td>
<td>• Yes for certain tasks, but not a widespread benefit.</td>
</tr>
<tr>
<td>Time zone efficiency</td>
<td>• Temporal distance allows Indian teams to confine meeting hours to a specific part of the day.</td>
<td>• Temporal overlap with two other sites does not allow Ireland to realise this benefit.</td>
<td>• Yes in India, not in Ireland. Unknown for US.</td>
</tr>
</tbody>
</table>
Pennysoft’s ultimate goal of their GSD activities is to reduce development costs. The data show that this is by far the most important benefit of GSD for them. This has lead the company to carry out software development activities at multiple locations in the US, Ireland and India. As such, the company has achieved its objective of access to relatively cheaper labour, along with cheaper real estate and infrastructure. GSD also allows Pennysoft to retain some software development skills in its home city on the US East Coast, giving direct access to their traditional corporate customers while attempting to reduce costs elsewhere. Skilled labour in Ireland and India has facilitated Pennysoft to move to relatively cheaper employment markets, although India currently lacks essential business expertise. Pennysoft’s flexible team structure allows for teams to incorporate skilled labour from multiple locations, leading to a more dynamic and fluid organisation.

However, rather than being able to leverage the large degree of temporal distance between team members, employees are forced to work during non-standard hours to maximise their temporal overlap with other sites. Both India and Ireland tend to work late, while the US is under pressure to start work earlier. The teams are also exposed to the difficulty of added overhead implied by having to coordinate across distances. This overhead is not taken into account by company policies which favour the up-front cost savings related to both Ireland and India when compared to the US. Ireland’s temporal overlap with both the US and India is used to mitigate coordination issues between them. However, Ireland’s managerial relationship with the rapidly-expanding Indian teams has led to a power struggle within the distributed teams. Emerging themes coming from these case study findings are addressed in the following sections.
5.2.15 Emerging theme: Site hierarchy

Section 5.2.2 described that there exists a hierarchy between the sites in Pennysoft. The US retains overall ownership of software projects. This is a natural progression, as software was being developed in the US before the company began to offshore to Ireland, and later to India. When the Irish site was being established, the US had been involved at a daily level in Ireland’s tasks. As the Irish site matured over the decade, teams there were able to take on more responsibility for larger sections of the work. Now with the introduction of the Indian site, the Irish case site closely manages their activities on a day-to-day level. This progression over time has led to a hierarchical structure of the sites, with the US at the top, followed by Ireland, and then India being at the bottom of the pecking order.

Several of the Irish-based interviewees commented on the problems resulting from the hierarchy between the sites, and in particular with the issue of managing the expectations of the quickly-growing Indian site. Interviewee B, an Irish project manager who closely manages the Indian resources within his team, felt that the India wanted to grow very quickly, as Ireland had done several years beforehand:

Something that I'm experiencing at the moment is that as India are coming up to speed... They want to grow very very quickly and go up the value chain like everybody else. (Interviewee B)

This drive for growth has caused tension within the hierarchy, where the Indian site wishes to gain more ownership of projects, and ultimately sees Ireland as an obstacle to its relationship with the US. Interviewee B was experiencing problems in that the Indian site was contacting the US customer directly saying “You don’t need to bother going through Ireland. We’re up to speed now, Ireland are great, but now we’re able to do what they do, why don’t you come to us”. The researcher had an interview with Interviewee B’s team lead in India, Interviewee C. This shed light on an interesting conflict in views. Interviewee C spoke more benignly of their relationship with the US:

There are some areas where we have built capability. In those cases, {Interviewee B} has told the US that they can directly contact me. (Interviewee C)

The opinions expressed by Interviewees B and C are conflicting. As such, the Indian and Irish sites have conflicting perceptions on the nature of their relationship with the US sites.
Interviewee H of Development Frameworks also experienced a power conflict with the Indian site:

Sometimes, India can be very protective... they’re trying to get work, trying to consolidate. They’re very competitive, and ambitious… And part of the problem was that the guys in India felt they owned it, it was theirs. And they resented anybody else. (Interviewee H)

Interviewee B concluded that the fix to this conflict in the hierarchy was to manage the expectations of the Indian teams. In effect, the Irish site is attempting to dampen the ambitious expectations that they perceive to be held by the Indian site. Both Interviewees F and K were of the opinion that more responsibility needs to be transferred gradually to the Indian site, ultimately resulting in a peer-based relationship between Ireland and India.

Table 5-4 examines the effects of this emerging theme on the potential benefits.
Table 5-4. The effect of the site hierarchy on the realisation of the potential benefits, where applicable.

<table>
<thead>
<tr>
<th>Applicable benefit</th>
<th>How the benefits are affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access to large skilled labour pool</td>
<td>Ambitious skilled workforce in India working counter-productively within the organisation, preferring to work directly with US, even if Ireland’s experience and temporal overlap would make their work more efficient. “I think that India are a bit naïve in that respect. They should use us. They should see the benefit of using Ireland.” (Interviewee B)</td>
</tr>
<tr>
<td>Cross-site modularisation of work</td>
<td>Tensions arise from Ireland’s need to “hand-hold” India, as India currently lacks business and technical experience, does not yet have the ability to work independently. If Pennysoft were to realise this benefit, it would help ease tensions by giving India better defined responsibilities.</td>
</tr>
<tr>
<td>Time zone effectiveness</td>
<td>India’s subordinate role in the hierarchy means that they accept the need to work non-standard working hours. Ireland also works as late as 8 or 9pm. This is in fact counter to the proposed benefit of time zone effectiveness. The flexibility in working hours by the distributed resources is of benefit for the company, as it means that their resources have a greater temporal overlap, which is preferred to a high degree of temporal distance.</td>
</tr>
<tr>
<td>Innovation and shared best practice</td>
<td>There is a lack of respect of the work ability of offshore sites. The US refused to accept processes agreed upon by Ireland and India. Best practice suggested by India not held in high regard by Ireland.</td>
</tr>
<tr>
<td>Increased autonomy</td>
<td>Potential benefit is affected by lack of experience in India, which in turn adds to tensions within the site hierarchy.</td>
</tr>
</tbody>
</table>

The conclusion that can be drawn from this example is that the benefit of access to a large skilled labour pool creates an imbalance of power and experience, with subordinate sites motivated to leap-frog others. While competition within the corporation may indeed be healthy, strong tensions between sites that do need to work together on a daily basis may by counter-productive. Interviewee B expressed that it was damaging for his organisation to have the customer see a high level of inner-fighting between sites. Ultimately in the researcher’s view, the company may need to modify the current balance of power between Ireland and India to avoid any potential damage to the company’s interests.

5.2.16 Emerging theme: Team structure adding to overheads

The case findings have shown that Pennysoft’s highly-distributed team structure allows teams to leverage skilled labour, wherever that labour is located globally. The company has
established sites in multiple locations in search of relatively cheaper labour. Therefore, we can conclude that the drive for reduced costs has lead to highly-distributed GSD teams in Pennysoft. However, interviewees expressed much dissatisfaction with the distributed context (in particular the Irish-based interviewees), with much effort required to coordinated resources across multiple locations. This coordination complexity may negatively affect the potential benefit of reduced development costs. Table 5-5 highlights the effects of distribution on the software development process, as stated by the interviewees.
Table 5-5. The effect of the cross-site team structure on the realisation of the potential benefits, where applicable.

<table>
<thead>
<tr>
<th>Applicable benefit</th>
<th>How the benefits are affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proximity to market and customer</td>
<td>Distributed team structure does allow for this benefit to be realised, with most team members being located away from the external customers.</td>
</tr>
<tr>
<td>Improved focus on documentation and improved process definition</td>
<td>Communication becomes more formal when distances are introduced (Interviewee G). Distribution encourages teams to create documentation and standardise processes, while this is seen as an overhead that would not otherwise be necessary.</td>
</tr>
<tr>
<td>Reliance on asynchronous communication</td>
<td>Must be used at times, but synchronous communication is still faster to resolve problems. Asynchronous communication across time zones can cause delays of 1.5 days.</td>
</tr>
<tr>
<td>Access to large skilled labour pool</td>
<td>Indian sites currently lacking business experience, requiring additional resources in Ireland to “hand-hold” them. Access to offshore labour pool increases overhead with daily developer-to-developer communication to provide help to less experienced developer.</td>
</tr>
<tr>
<td>Access to cost-effective labour pool</td>
<td>As noted above, more resources are required to manage the newer teams, thus negating up-front cost savings. Team members seen as “widgets” by the accountants who decide on which sites should be involved, but they do not recognise the considerable friction costs.</td>
</tr>
<tr>
<td>Cross-site modularisation of work</td>
<td>The lack of modularisation of work tasks in itself requires tighter coordination across sites. Sites are coordinating on a daily basis, having to deal with the distances between them, indicating a high degree of daily project management overhead.</td>
</tr>
<tr>
<td>Time zone effectiveness</td>
<td>Distributed teams are generally not realising this benefit. Temporal distance is a barrier to coordination within the distributed team structure. Ireland is acting as a go-between for the US and India which helps to alleviate problems due to temporal distance. However, Irish-based interviewees expressed dissatisfaction with being over-worked due to holding this role.</td>
</tr>
<tr>
<td>Time zone efficiency</td>
<td>Again, the team structure is forcing the temporal distance to be minimised, rather than leveraging elements of temporal distance.</td>
</tr>
<tr>
<td>Increased autonomy</td>
<td>Cross-site team structure means that there is tight cross-site control mechanisms, reducing the possibility for site autonomy.</td>
</tr>
</tbody>
</table>

This emerging theme of a high degree of overhead resulting from the distributed team structure was highly evident with Irish-based interviewees. In particular, the interviewees at
project management level were expressing dissatisfaction with dealing with both the Indian and US sites during their working day, being “caught in the middle” (Interviewee B). These managers described their mornings in dealing with overnight e-mails from the Indian site, then speaking with the Indian site on the phone, and then the US site begins their morning. At this point, the US employees begin to deal with e-mails during their morning commute. Ireland deals with these e-mails, and there may be afternoon teleconferencing meetings with the US. It is clear from the findings that the Irish site is straining to deal with the heavy communication requirements of the distributed team structure. The next section discusses how this added overhead is not taken into account when estimating the costs of projects.

5.2.17 Emerging theme: Costing of GSD projects

As already emphasised, the interviewees viewed Pennysoft’s ultimate goal of their GSD operations to be to develop software at reduced cost. Pennysoft is a financial company, which according to Interviewee H accentuates their focus on reducing costs, however that may be achieved. As previously quoted, Interviewee H felt that the accountants making decisions on where to develop software do not understand the process of software development and how distances can lead to inefficiencies. At the very least, this indicates a lack of visibility or transparency of corporate policy with respect to the middle-management interviewed.

Pennysoft’s policy of requiring 40% of project resources to be offshore in effect mandates their software development activities to take place in a distributed setting. This can be best seen in the example of the Development Frameworks team, where they had a choice of either decreasing the project headcount or taking on two resources in India. The latter was decided upon, resulting in a team of fourteen people across six locations in three continents. The data has shown that this cross-site management of resources of varying experience requires dedicating considerable time and effort to sustain. Interviewees expressed the opinion that this has resulted in higher-cost resources in Ireland devoting time to work through problems with their Indian counterparts. This added requirement for dedicated resources seems counter-productive to the expected benefit of cost savings. It is a significant finding that the company’s drive toward GSD does not take into account the complexity involved with developing software in a distributed environment and the related costs of such an endeavour.
Pennysoft do not have a costing model for GSD which takes into account these additional overheads, when compared to co-located software development. The cost of leveraging a cheaper offshore workforce has seemingly not been taken into account. Rather, decisions have been taken based on the direct cost-per-hour of each site. Finally, Interviewee I noted that upper management was beginning to recognise the difficulties related to globally distributed teams. The interviewee was in the process of attempting to cut back on the number of sites involved in his fourteen-person team, so as to re-locate work from India to Ireland.

5.3 Case 2: Semicon

Here, the empirical findings of the Semicon case study are presented. First, the organisational context of the case study site is presented. Second, the team structure opted for by Semicon is described, as this affects the very nature of their GSD activities. Third, the extent of realisation of the potential benefits by Semicon are described through the findings of the case. Fourth, within-case analysis begins with a summary of the empirical findings with respect to each benefit. Within-case analysis is offered based on themes which emerged through analysis of the empirical findings.

5.3.1 Organisational context

The Irish case study site is a key site for the company. The general manager of the Semicon Communications Europe sub-company (Interviewee A) resides at this site. This sub-company is managed within the larger ‘Semicon Digital Enterprise Group’. This general manager manages the software engineering organisation of this group, and is comprised of 400 people across eight sites. The general manager directly manages three managers in Ireland, two managers in India, one manager in Malaysia, and five managers in the US.

Software development projects typically consist of 50 people, and last one year. Several business units will be involved in such projects, including the finance, marketing (both hardware and software), project management, software, silicon, and manufacturing business groups. Semicon has developed the concept of ‘core teams’, where a core team leads each project, with the business units involved reporting to that core team. The Irish case study site feeds into these core teams by representing the software group. An
abstracted overview of the organisational structure of the software group is illustrated in Figure 5-5.

**Figure 5-5. Illustrative example of Semicon's software program structure.**

The Irish-based program manager shown in Figure 5-5 is responsible for 400 employees across eight locations including North America, Ireland, India, Poland, China, and Malaysia. Each site has one or more software development teams, and typically has one project manager. Each project manager manages one or more teams at their site. Then, each team is led by a team lead who is responsible for the technical implementation of the requirements. Several such software development teams are located at the Irish case study site.
The software that the software business unit develops is mostly free of charge: the software “enables” its hardware products. For example, software drivers are developed to allow its hardware capabilities to be exposed to third-party vendors. Software requirements come from the marketing unit and from particular customers. The software program manager negotiates a set of requirements with the marketing team.

Projects covered by the Semicon case study varied in both size and duration. For example, one project involved developing a tool suite and lasted for five years. Another project developed a source code management tool, and lasted for seven months. Each of the projects is described here:

- The **Network Processor project** dealt with Ethernet acceleration, telephony, and developing access layers between the hardware and external software. The interviews took place two weeks after the project had been prematurely cancelled. It had been running since 2004, but its outcome was no longer required by the company. Software development spanned three locations in the US (8 people), a team of developers in Ireland (16-19 people), and Taiwan (3 people). A team in Bangalore had been coming on-board, and the project had close collaboration with a separate team in Malaysia.

- The **Software Development Kit (SDK) tool suite project** was developing a tool suite (including compilers and debuggers) to support the Network Processing division. It was a large project that had been running since 2001. During data collection for the case study, the project was in maintenance phase. The project was located across two sites in the US (5-6 people), Ireland (11 people), Poland (5 people) and India (7-8 people, increasing to 10 people for maintenance phase).

- The **Source Code Management Tool project** developed an in-house tool for software engineering, and lasted for seven months. The project was located across site in the US (1 person, plus several peripheral team members), Ireland (3 people), India (2 people), Malaysia (3 people) and Poland (several peripheral team members). The peripheral team members did not dedicate all their time to the project, but rather offered technical assistance.

Table 5-6 gives a complete list of interviewees who were involved in the case study. All interviewees of this case study were based in Ireland.
### Table 5-6. Semicon interviewee codes, projects, and roles.

<table>
<thead>
<tr>
<th>Project/Interviewee</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Management</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>General manager of software sub-company.</td>
</tr>
<tr>
<td>B</td>
<td>Staffing manager</td>
</tr>
<tr>
<td>C</td>
<td>Project and integrations manager</td>
</tr>
<tr>
<td>Network Processor</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Project manager</td>
</tr>
<tr>
<td>E</td>
<td>Technical lead</td>
</tr>
<tr>
<td>F</td>
<td>Team lead</td>
</tr>
<tr>
<td>SDK</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Technical lead</td>
</tr>
<tr>
<td>H</td>
<td>Team lead</td>
</tr>
<tr>
<td>Source Code Management</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Software engineer</td>
</tr>
</tbody>
</table>

### 5.3.2 Team structure

Semicon – especially at higher levels of management interviewed – are conscious about their teams’ structure and how that affects the distributed software development process, as discussed in following sections. Figure 5-6 illustrates the typical structure of a team in Semicon. In the context of Semicon, the definition of a ‘team’ is somewhat different to Pennysoft. A team is defined in Semicon as a sub-unit of a project, and each team has its own internal organisational structure. Multiple teams cooperate to produce the software required for a particular project.
Figure 5-6. Semicon team structure.

Figure 5-6 may seem simplistic and perhaps even unnecessary, but this team structure is important in the context of GSD. Semicon designs each team to be a coherent co-located set of individuals. That is, all team members are always located at the same site. Before this case study had taken place, Semicon had had a more flexible team and project structure. Software activities were highly-dispersed across the eight software development sites. Interviewee B had direct interaction with those sites:

*It isn’t good for your sleep! I was an EPL [engineering product lead] dealing with eight sites globally. Time zones are the biggest problem when organising the different parties in a project […] We know [cross-site] communication is inefficient. (Interviewee B)*

The same interviewee even joked that they were having communication issues across buildings at this Irish site, as the group was in the process of moving from one building to another in the same industrial park. Interviewee B said that managers at Semicon made a conscious decision to limit the distribution levels of their software development projects to avoid the difficulties of having to work across geographical and temporal distance. This decision was highlighted by the general manager of the software program (Interviewee A). Managers decided to limit all software development projects to only include teams from two “geographies” or “geos” at a time. A “geography” is their loose description of a region covering a minimal range of time zones and within the same continent. This policy being enforced by Semicon does not set a hard limit to the number of sites that can be involved.
in one project, but it is driven by management’s belief that software development is made easier if the number of cross-site and cross-time zone dependencies can be reduced.

Hierarchy

Interviewee B saw value for the Irish site having the software program management being located there:

>This is a huge value-add for us here in [Ireland]... we are recognised with delivering whole projects. It is very important for us politically to be seen as this role compared to other sites. We try to be as high-profile as possible. (Interviewee B)

In a way, the geographical separation of the teams allows for the Irish site to distinguish itself from other sites as a highly-valuable organisational unit. Apart from the program management, teams within a project were described by interviewees as being peers (this is the view of Irish-based interviewees, perhaps other sites have differing perceptions of this organisational structure). Interviewee C stressed when asked about the role of the Indian team that they are a “peer team rather than a sub-team”:

>Bangalore don’t want to be second in line to a small location like Ireland. (Interviewee C)

This comment illustrates that there are tensions between teams, but that teams are not being dominated by other superior teams elsewhere. There is also competition between teams to be able to attract the most valuable work, where teams “grab the best work” at requirements stage (Interviewee C).

Inter-team and inter-site communication

Further to Semicon’s effort to limit the level of distribution in projects is their stated aim of reducing inter-dependencies amongst the teams. Interviewee expressed the opinion that teams with fewer inter-dependencies can act more independently and are less inclined to communicate across the geographical and temporal distances, which in their past experience is costly. Interviewees A and B spoke enthusiastically of attempting to achieve this through designing “loosely coupled and highly cohesive teams”. Each team typically works on a defined portion of functionality of a software development project. Their aim is “to have
“nothing to do with each other” (Interviewee B), thus avoiding more costly cross-site communication.

Semicon software managers are very conscious of who talks to whom across geographical and temporal distances and whether that communication is effective, worthwhile and necessary. Still, cross-site communication does continue as they cannot completely eliminate it. See Figure 5-7 for a demonstration of the typical inter-team communication structures.

**Figure 5-7. Typical structure of inter-site communication between teams in Semicon.**

Figure 5-7 demonstrates several types of inter-team communication during the software development process:

- *Formal planned communication* regularly happens between project managers, team leads and key developers, in the form of weekly project status meetings. This is the main form for the teams to synchronise their efforts. The teleconference meetings can include the use of an electronic ‘dashboard’ which allows for the sharing of important figures and documents during the meeting. There is a greater emphasis
on these meetings during the latter stages of the software development process when deliverables are being prepared.

- **Unplanned inter-team communication** happens between the team leads across sites as required. Communication is passed through these team leads in order to control the amount of unplanned inter-team communication. Such communication can be daily.

- **Inter-developer communication** happens depending on the phase of the project for dealing with detailed problems “*when it’s efficient and practical to do so*” (Interviewee B). Inter-developer communication is the most undesirable communication structure for Semicon. They recognise that it is difficult and inefficient to try to solve technical problems across geographical and temporal distances. Team leads monitor inter-developer communication across teams in order to “*not let the amount of communication cause chaos*” (Interviewee B).

The following sections describe the realisation of the potential benefits of GSD as in the findings of the case study.

5.3.3 Proximity to market/customer

As Semicon is primarily a manufacturing company in the telecommunications and electronics industries, many of its customers are located in Asia, and in China specifically. By locating software development activities there, Interviewee A stated that it has been able to improve direct relations with customers there. Having native Chinese teams allows the company to be not only geographically closer to its customers, but also closer to them linguistically and culturally. This benefit was noted by the software program manager. In particular, Interviewee A noted that Semicon’s US West Coast culture was at odds with Chinese culture, where “*everything is done by consensus*”. Customer-facing resources in China mitigate the interactions between customers and the US-based business units.

*Conclusion:* Semicon is realising this benefit of GSD.
5.3.4 Improved focus on documentation

Interviewees did not strongly agree with this potential benefit of GSD. Rather, they expressed that it was being realised due to the heavy technical nature of their engineering work. Formal communication is indeed a fundamental part of Semicon’s software development activities. Semicon projects focus much energy on the initial and final stages of projects, which is further described in the following sections. The core team lays out initial documentation in line with its agreed requirements in order to communicate to the different teams. This is, however, in the context of extensive travelling by key engineering staff at the initial phase of a project. It also does not rule out the possibility of misunderstandings:

“There can be different interpretations of a feature requirement. We must keep a check on this.”
(Interviewee G)

However, Semicon teams are co-located units, and so the geographical, temporal and socio-cultural distances do not play an essential role in creating team-centric documentation.

Conclusion: this benefit is not attributed to GSD, although there is heavy influence on documentation within Semicon.

5.3.5 Record kept of communications

For the individual software development teams, written formal communication is used to communicate detailed requirements at the beginning of a project. Generally only project managers, and sometimes their team leads, communicate with other sites across temporal distance. Even then, there is a preference in Semicon for planned formal synchronous communication rather than unplanned asynchronous communication. For example, Interviewee I felt that written communication was to be kept formal:

“Email is a step up in formality. If you have a gripe about whatever reason, you could mention it to an employee at the next desk, but you wouldn’t mention it in email.”
(Interviewee I)

In line with general expressions throughout the Semicon interviews, this quote hints to us that informal cross-site communication is kept to a minimum. Weekly meetings are the primary medium of cross-site communication. There are processes in place to record
actions from these meetings and to distribute them to all meeting participants, thus creating a record of the communication. When asked on this potential benefit, interviewees did not relate to it, as synchronous formal communication was generally preferred.

However, despite the preference for synchronous communication, Interviewee C noted that 80 percent of his communication with the Indian team was through e-mail despite a temporal overlap during Ireland’s mornings which would allow for more direct synchronous communication. Interviewee E, of the Network Processor project, noted that e-mail is not enough to maintain shared context between the teams working on a project. Instead, the interviewee noted that it must still be accompanied by planned synchronous communication.

Conclusion: Not accepted as a particular benefit of GSD. Detailed documentation is produced in line with the organisation’s strong engineering culture, and teleconferencing meetings are held to synchronise efforts.

5.3.6 Access to large skilled labour pool

The Irish site manager was enthusiastic of the possibilities offered by locating in different countries in the pursuit of highly-skilled engineers.

*Intellectual horsepower is a major benefit of GSD for us. We recruit from the top four or five universities in India. For each place in the university, there are something like 40,000 applicants. We can pick from the resulting Ph.D. students, giving access to genius employees… They're really the cream of the crop. (Interviewee A)*

As Semicon is a knowledge-based company, it is beneficial for the company to hire new talented employees, wherever that employment market may be (but preferably in lower-cost employment markets). Some existing sites may simply not have the capacity to take on additional work due to a limitation on the size of the skilled labour market available to them.

This benefit of access to large skilled labour pool, however, is eroded by concerns over the rate of attrition in the organisation. Interviewee C, for example, had “heard rumours of 50% attrition in Bangalore”. A high rate of attrition occurs in the Bangalore software employment market due to intense competition for talent there. Employees with sufficient technical
skills are quite difficult to come by for Semicon. When an employee leaves the organisation, they must re-invest in searching for a replacement and bringing them up to speed over the course of several months. This means that a high attrition rate for Semicon translates into considerable increased costs. Attrition can also be of concern when dealing with third-party vendors, a problem that Semicon does not have full control over:

_Offshore sourcing vendors are still dollar-focused… In a way, a high turnover is encouraged. The work is tough, and after two years’ work, the company would favour newer younger replacements to take an employee’s place._ (Interviewee C)

It emerged during interviews that the issue of a high rate of attrition may also be attributed to the nature of work being sent offshore. Interviewees indicated that Semicon has a tendency to ‘keep the core and offshore the chore’. That is, interviewees hinted at the fact that Semicon’s core architecture work is probably performed at their headquarters in the US:

_We won’t offshore the crown jewels._ (Interviewee F)

Interviewee H stated that “the US probably gets the new exciting projects”.

We can therefore see a tension between the company’s perceived benefit of access to “the cream of the crop” employees, and the work that is being assigned to them in practicality. The researcher speculates that if “genius” employees are given quite menial tasks, then we may expect morale to be reduced and rates of employee attrition to increase.

**Conclusion:** Semicon actively seek out talented labour, but may be limiting the extent of realisation of this benefit by not sending offshore the most interesting work of working on the “crown jewels”, or the core functions of the company.

### 5.3.7 Access to cost-effective labour pool

When asked why Semicon has distributed its software development, cost savings was the primary reason given by every interviewee, above that of seeking large skilled labour pools. They recognise that sites in countries such as Poland, Malaysia, India and China are involved in Semicon’s GSD efforts primarily due to cost savings:
There is a general shift towards low-cost geos. Otherwise, all the work would be done in the US. (Interviewee H)

The software program manager offered general cost saving guidelines based upon salaries and infrastructure. Project costs in India are three times cheaper compared to Ireland. Compared to the US, Indian projects are four times cheaper. In that case, Ireland is 25 percent cheaper than the US. Interestingly, the Irish site was acquired by Semicon partly due to relatively low wages there. However, the interviewee declined to offer actual cost figures. Table 5-7 summarises the per-quarter cost of the US, Ireland and India.

Table 5-7. Approximate rate per hour, as charged by each site to internal customers. Assume x to be the per-quarter cost charged by India.

<table>
<thead>
<tr>
<th>Location</th>
<th>Rate per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>4x</td>
</tr>
<tr>
<td>Ireland</td>
<td>3x</td>
</tr>
<tr>
<td>India</td>
<td>x</td>
</tr>
</tbody>
</table>

Project cost calculations are based on the per-quarter fee of each team being considered for inclusion in the project. Semicon also make limited use of outsourcing to third-party vendors in order “to keep the costs down” (Interviewee B). An outsourcing manager is appointed at the Irish site to deal with those vendors.

It should be noted that there can be a difference between the reasons a company set up a new site, and factors that are taken into account when deciding which teams and sites to include in particular projects. A company may be motivated to set up a location due to factors such as relatively lower wages, available infrastructure, availability of skilled workers, and the cost of real estate.

However, when a project core team is deciding which teams and sites should be involved, the per-quarter billing cost of each team is taken into account, along with other factors such as: the headcount and number of staff-months required for the project, and the headcount available at each site along with their skill sets. Interviewee A noted that for less complex work, the site of lowest cost is given the work. However, in more complex skilled work, the cost charged by sites is not necessarily the primary selection factor. Interviewee
H noted that for his SDK project, the maintenance phase had been handed over to India because probably because it was a lower-cost location.

As a side-effect to Semicon’s drive to conduct software development in relatively cheaper economies, some interviewees noted that it may have some political implications within the company, insofar as employees in higher-cost locations may be cautious about the possibility of working with lower-waged colleagues. Interviewee F noted that offshoring must be handled carefully when the policy is being presented to employees. The consolation for the employees in higher-cost locations is that their expertise is still valuable.

Interviewee C felt that large corporations, including Semicon, have no loyalty to particular sites. If another location can offer a better value proposition, then work will be moved there. Interviewee D noted that when there are cost-saving measures in place, travel between sites is the first thing to be cut.

Conclusion: It seems that this benefit is being realised by Semicon. However it is not clear from the data if there are added overhead costs for managing the distributed projects. The company certainly invests much time and effort in the initial and end phases of projects so as to reduce coordination costs while software is being developed.

5.3.8 Cross-site modularisation of work

It has already been shown that Semicon strives towards “loosely coupled and highly cohesive teams”. The aim of loose coupling is to minimise the number of dependencies between teams. High cohesion of a team’s workload means that all work tasks are focused and closely related. To allow for the teams to be loosely coupled while working within the same overall project, Interviewee B stated that the responsibility of the development of defined parts of the system must be decided upon. Managers at Semicon in effect ‘de-couple’ the teams from each other by assigning each team a specific set of features to develop. Features are both designed and assigned to teams while keeping in mind the features’ dependencies on others. The stated aim of this approach is to minimise dependencies between teams during the software development activities, with an ultimate aim of reducing the costly effects of cross-site communication and coordination:

Because of feature splits, we only need to contact the other site maybe once or twice a week, outside of management meetings. (Interviewee H)
This development approach by Semicon requires a great deal of planning at the initial stages of projects. It requires them to define in detail the dependencies and interactions between each element in the system, the system’s components are then assigned to different teams. A side-effect of the approach is that much effort is also required at integration stage (when the system’s components are merged to create a working product). This is due to misinformed assumptions taken by the teams, despite the planned formal communication throughout the project. The integration phase involves re-aligning all system components so that they can be integrated without introducing coding errors. Managers and some key developers travel for both the initial and integration phases. As the planning and integration phases require intense coordination, face-to-face meetings are preferred despite the cost of travel.

This forward-planning approach by Semicon means that the set of requirements must be kept stable once decided upon. A change of requirements would result in re-synch meetings between all stakeholders to assess the impact the change has on their dependencies, and then changes to their work plans. A change of requirements would have to be accepted by all involved:

A site can enter a change request if they require a change in the project. Marketing will ultimately decide if the change is needed. Marketing will ask the site for an assessment of the impacts it would have on other work. This is carried out on a program-level. Also, we need to meet again if there’s a re-mapping of requirements. (Interviewee H)

There is an understanding across the organisation that requirements must generally remain stable, and this seems to be ingrained in their organisation culture; it is “extremely difficult” to have a change agreed upon (Interviewee B). That being said, one interviewee noted that the nature of Semicon’s hardware industry means that the hardware section of the product takes a long time to design and to develop. This puts less pressure on the software-producing teams, and helps requirements to remain stable:

Modularisation of the system and team structure is perceived as a benefit to Semicon in that each component from each team can be deployed to the customer as each component becomes available.

Blocks of software are built in each site, and then the blocks fit together in the end. Individual releases can be made available to the customer without every block being prepared. (Interviewee I)
The desire for re-use of code across teams is one factor that works against the modularisation of work tasks. Code re-use requires inter-team coordination in order to decide what code can be re-used, but its ultimate purpose is to reduce cost. Interviewee B spoke about this issue, and did not seem certain on how to balance code re-use with decoupling teams:

"This is our paradox: our teams' structure means that we want to be as loosely coupled as possible, but we also want to have re-use as much as possible. The loose coupling isn't a perfect picture. (Interviewee B)"

**Conclusion**: this benefit certainly is being realised by Semicon, following previous bad experience with software development projects being overly distributed, making the coordination effort involved too difficult. The overhead to this approach is that there is intensive effort for the planning and integration phases, whereas software development during the project can be performed at each team with relative independence.

### 5.3.9 Improved process definition

Similarly to the potential benefit of improved focus on documentation, interviewees felt that Semicon was a process-mature company with its focus on engineering. This benefit is related to the previous benefit of cross-site modularisation of work. Mature processes allow the company to distribute their software development activities.

**Conclusion**: not perceived to be a benefit of GSD, as such, as the company enforces a strong engineering ethic.

### 5.3.10 Time zone effectiveness

Follow-the-sun development typically involves having a hand-off of work tasks between teams in three different time zones every eight hours. Such a model involves teams being completely dependent on the work of colleagues in other locations. Semicon’s approach of loosely-coupled co-located teams and relatively low communication levels between them is in fact **incompatible** with the follow-the-sun model. The follow-the-sun model involves a high degree of inter-team dependencies, tasks being performed across several sites, and with daily synchronisation of tasks. In contrast, Semicon’s approach involves a low degree
of inter-team dependencies, tasks performed within one site, and weekly formal synchronisation meetings.

Interestingly, in relation to this potential benefit, Interviewee C stated that software development is a long-term process which should be planned over a number of months rather than requiring daily coordination and hand-offs. Interviewee B said that “[follow-the-sun] would not be practical for software development. Developing software at one site, and testing it during the night at another site would be more practical”.

**Conclusion:** the benefit is not being realised, as it is at odds with Semicon’s approach to software development.

### 5.3.11 Time zone efficiency

Again, this benefit is not of relevance to Semicon’s software development activities, as temporal distance between sites is not being leveraged, but rather its effect on daily development activities is being minimised.

**Conclusion:** not being realised due to the low degree of coordination levels between teams across temporal distance.

### 5.3.12 Innovation and shared best practice

Interestingly, this is another potential benefit that may be at odds with Semicon’s approach to limit informal communication between sites. The researcher speculates that informal communication may be one way for innovation to freely spread between sites. Interviewees disagreed that best practices flow between sites in Semicon through informal means. They said that teams located at the same sites were better able to facilitate such transfer of best practices as there was close informal contact between the members of the different teams. This may potentially be limiting innovating ideas coming from the “genius” employees that the company seeks to employ.

While informal means of sharing best practices are limited, interviewees agreed that there are corporate policies in place to identify and share best practices between teams. However, no single interviewee strongly expressed the opinion that their team was benefiting from best practices coming from other teams, arising from any aspects of GSD.
Conclusion: this potential benefit is being promoted through formal methods, but it does not seem to be making noticeable changes within teams.

5.3.13 Allocation of roles and team structure

Semicon realises this benefit at an organisational level – project teams can be either internal or external. External teams are outsourced to other organisations that feed in to Semicon’s corporate ‘ecosystem’. Interviewee B was the only interviewee to discuss this potential benefit in relation to inter-team flexibility in allocation of roles:

In our project, Bangalore is taking on all the work… There is a hire freeze at this [Semicon] site. Therefore offshoring work frees up employees in Ireland to perform other work. (Interviewee B)

In this context, the team structure seems to be flexible enough in order to allow reallocation of work to balance the work load between teams. However, a majority of interviewees did not agree that this potential benefit was being realised, as the corporate culture is perceived to be supporting stability in the organisational structure. This means that teams are initially selected to work in a certain project, and after that the teams will focus on completing that project for a month or longer. The organisational culture in Semicon does not favour quickly reallocating work from one team to another.

Conclusion: The benefit is realised in that entire teams can be sourced externally, and Interviewee B points towards the re-allocation of work between teams. However, the overall corporate culture tends to work against the realisation of this potential benefit.

5.3.14 Increased autonomy

Interviewees generally agreed that reduced inter-team dependence allowed for each team to define its own working culture. Individual teams can set their own work practices, including processes and tools as needed. It seems that teams were quite satisfied with this structure, as it gave them some independence from other sites. The realisation of this benefit is as a result of the independence offered through cross-site modularisation of tasks.
There is a constant tension between teams attempting to adopt their own internal practices, while ensuring that their output is compatible with other teams’ components. Integration phase brings to a head the difference in approaches adopted by different teams:

*During integration phase, when putting two features together, there can be unexpected behaviour.*

(Interviewee G)

Interviewee B also discussed the measures taken to promote code re-use amongst teams. Such re-use is being promoted within Semicon as a way of reducing time-to-market. However, it places a limit on the extent to which teams can develop individually, as they must put in place processes to ensure that the code being developed is not yet available within the company’s code libraries.

**Conclusion:** Semicon teams by their nature have the ability to code independently of other teams, as their dependencies on other teams are limited. However, it is not possible to fully break away from other teams’ development activities. There is pressure to produce standardised code that can be integrated with other teams’ work, and teams must ensure that they maximise code re-use of existing code.

5.3.15 Within-case analysis: Evidence and drawbacks of the benefits

Table 5-8 summarises the benefits that Semicon realises due to the distributed context of their software development work.
Table 5-8. The realisation of the benefits of GSD at Semicon.

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Evidence</th>
<th>Challenges</th>
<th>Realisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proximity to market and customer</td>
<td>• Geographically and culturally closer to hardware-focused customers.</td>
<td>• Introduces additional distance to contend with.</td>
<td>• Yes.</td>
</tr>
<tr>
<td>Improved focus on documentation</td>
<td>• There is a strong emphasis on documentation.</td>
<td>• Not attributed with GSD, but rather to Semicon’s engineering traditions.</td>
<td>• No, not accepted to be a GSD benefit.</td>
</tr>
<tr>
<td>Reliance on asynchronous communication</td>
<td>• Much asynchronous communication does take place.</td>
<td>• Record kept by e-mail not perceived to be a considerable benefit.</td>
<td>• No.</td>
</tr>
<tr>
<td>Access to large skilled labour pool</td>
<td>“Cream of the crop” graduates.</td>
<td>• High attrition levels reported.</td>
<td>• Yes, but perhaps not maximising their access to “genius” employees.</td>
</tr>
<tr>
<td>Access to cost-effective labour pool</td>
<td>• This is the driving benefit of GSD.</td>
<td>• Unclear as to whether overheads are being accounted for.</td>
<td>• Yes, but much effort to break cross-team dependencies for GSD.</td>
</tr>
<tr>
<td>Allocation of roles and team structure</td>
<td>• Entire teams can be externally sourced.</td>
<td>• Corporate culture frowns upon re-allocations during projects.</td>
<td>• Yes, at project rather than team level.</td>
</tr>
<tr>
<td>Cross-site modularisation of work</td>
<td>• Loosely-coupled teams reduce coordination complexity of cross-site communication.</td>
<td>• Difficult to implement, still need for some unplanned inter-site communication.</td>
<td>• Yes, with significant effort.</td>
</tr>
<tr>
<td>Improved process definition</td>
<td>• Process-mature teams.</td>
<td>• Not attributed to GSD, but rather the engineering tradition of the company.</td>
<td>• No, not perceived as a benefit of GSD.</td>
</tr>
<tr>
<td>Time zone effectiveness</td>
<td>• No evidence.</td>
<td>• Not relevant to Semicon’s approach.</td>
<td>• No.</td>
</tr>
<tr>
<td>Time zone efficiency</td>
<td>• No evidence.</td>
<td>• Not relevant to Semicon’s approach.</td>
<td>• No.</td>
</tr>
<tr>
<td>Innovation and shared best practice</td>
<td>• Corporate processes encourage identification and sharing of best practices.</td>
<td>• Lack of cross-site informal communication limits natural sharing of best practices.</td>
<td>• No particular evidence of this benefit in practice.</td>
</tr>
<tr>
<td>Increased autonomy</td>
<td>• Geographical separation implies teams’ ability to work at their own pace.</td>
<td>• Tension between increased autonomy and the necessity of maintaining communication with other teams.</td>
<td>• Yes.</td>
</tr>
</tbody>
</table>

Semicon are certainly achieving some of the benefits of GSD, but it is not always clear-cut on whether a benefit is being realised to any significant extent, and whether the indicators
of the benefit are due to the nature of GSD. Semicon seeks additional skilled labour in relatively cheaper markets such as in Bangalore in India and Penang in Malaysia. This leads them to modularising their work tasks according to system components, but inter-team communication still does take place, and the planning and integration required to have loosely-coupled and highly-cohesive teams may be almost prohibitive. Semicon’s approach rules out benefits of leveraging temporal distance, and they at odds with the sharing of innovation of best practices. The following sections analyse the emerging themes of this case’s finding.

5.3.16 Emerging theme: Team structure to reduce coordination overheads

The potentially negative effects of geographical and temporal distance on communication are of much concern to Semicon. As previously quoted, “we know [cross-site] communication is inefficient” (Interviewee B). Semicon teams do suffer from the effects of cross-site communication, but the problems were greater in the past. Project managers sometimes still need to hold “share the pain” meetings involving three continents, where the timing of the meeting may not particularly suit any of the teams (Interviewee A).

Because of the coordination complexity involved with cross-site communication, Semicon has attempted to structure their projects so as to minimise cross-team interdependence. According to their accounts, this involves meticulous design of the system being developed so as to allow a team to develop a system component in isolation. Semicon teams are co-located, so that day-to-day development work generally only involves colleagues who are located at the same site. The approach is not perfect, with some inter-site dependencies still remaining. Complete task de-coupling may never be achievable because each team’s work will depend on other components being developed by other teams for the same overall project, requiring coordination to manage the dependencies.

One side-effect of this team structure and its focus on reducing communication levels between teams is that the communication that does take place is more formal (Interviewee B). Cross-team meetings follow a set agenda which is sent to all participants. The decisions taken during meetings are also sent to all participants. There is emphasis on inter-team processes and heavy documentation. Interviewees noted that the culture of formalisation of communication is promoted explicitly through the company’s policies.
By analysing the findings that managers at Semicon have chosen the modularisation of work tasks rather than having teams that freely communicate across continents, time zones, and cultural distances. The approach to cross-site modularisation of work requires significant up-front investment. This up-front project planning involves identifying all potential system dependencies, and how those dependencies may affect the development teams’ dependence on each other. Such planning is very detailed and requires face-to-face discussion. Managers and key developers travel to one site to plan for the project.

The approach also requires travel by managers and key developers at integration phase. Interviewees B and G noted that the loose coupling of teams can result in “big-bang” integration, where each team’s unique interpretations of the feature requirements clash when integration is attempted. Interviewee B suggested the need to limit the effect of big-bang integration is one reason why inter-team communication cannot be completely eliminated.

Overall, Semicon’s approach to modularise work tasks introduces, or at least increases, overheads related to the cost and effort required to allow for management and key developers to travel. Travel required for integration can be time-consuming. For example, interviewees pointed to the difficulties of travelling to Penang, Malaysia, and the difficulty of three connection flights in order to reach Gdansk in Poland. Visa requirements must also be planned for well in advance. Table 5-9 examines the effects of this emerging these on the potential benefits.
Table 5-9. The effect of the team structure on the realisation of the potential benefits, where applicable.

<table>
<thead>
<tr>
<th>Applicable benefit</th>
<th>How the benefits are affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proximity to market and customer</td>
<td>The team structure is an enabler of this benefit, it facilitates GSD, and so makes it somewhat easier to maintain offshore teams.</td>
</tr>
<tr>
<td>Access to cost-effective labour pool</td>
<td>The creation of loosely-coupled teams requires much effort for each project. Of question is whether the added effort still makes it worthwhile employing cheaper offshore employees.</td>
</tr>
<tr>
<td>Allocation of roles and team structure</td>
<td>This team structure is rigid, and only includes team members from one site. This does not accommodate easy shifting of labour between teams by changing team boundaries to include labour from other sites.</td>
</tr>
<tr>
<td>Time zone effectiveness and efficiency</td>
<td>The team structure renders these potential benefits redundant.</td>
</tr>
<tr>
<td>Innovation and shared best practice.</td>
<td>Semicon actively seeks ways in which to reduce the need to communicate informally across sites. This negatively affects potential sharing of innovation, as there is not a natural flow of information between sites.</td>
</tr>
</tbody>
</table>

The company has previously experienced problems with less structured projects. We can take from this that the perception of Semicon is that the benefits provided by modularisation of work out-weight the challenges involved, and that these challenges are not as damaging as the direct effect of the distances on the software development process.

5.3.17 Emerging theme: Costing of GSD projects

While the last section has concluded that Semicon favour the effort required to modularise work tasks compared to the coordination costs of looser team structures, it is not clear to what extent this effort affects the realisation of the benefit of reduced development costs. The extent of this effort is considerable, the effort needs to be carefully managed, while assessing the related costs and managing problems with cultural distance (Interviewee C).

Interviewees confirmed that the up-front cost of travel is included in the costing of projects. Interviewee A noted that developers have travelled for up to 6 month periods to other sites to train them in, saying that “it’s essential that developers travel”. Such travel can be expensive. A casualty of the company’s aim to reduce costs is a tight restriction on travel, particularly for developers. Interviewee H stated jokingly that “you could say that our levels of
travel change directly in relation to our stock price”. Interviewee E noted that it’s difficult to be authorised to travel to Penang, Malaysia. Interviewee G noted that to travel from Ireland to the development site in Poland requires three flights. On one trip the interviewee missed a flight and the trip took 14 hours.

However, the importance of face-to-face meetings is accepted, and managers and key developers travel both at the initial and end phases of a project (although most developers never having met remote colleagues). Cross-site communication requires communication technologies such as telephone, e-mail, instant messaging and teleconferencing. However, these technologies do not fully make up for the loss of face-to-face contact which would take place in a co-located setting:

You can’t beat face-to-face… There has been research in [Semicon] suggesting that 80% of face to face communication is non-verbal. (Interviewee H)

I found one person in Poland who I was working with remotely to be 'gruff’. But after meeting with him face-to-face, I had a much different perception of him. (Interviewee I)

5.3.18 Conclusion of case findings

Overall, it is evident that Semicon’s approach to GSD is not cost-free. It involves much up-front effort including the travel of managers and key developers. Interviewee C summed up his impression of the value of GSD:

In essence, GSD is not easy. If I were setting up a team on my own in my own company, offshoring work is the last thing I would want to do. You want to keep software development all in one place. That’s the way to success. (Interviewee C)

Semicon realises that GSD itself is not an ideal context of software development, and that co-located software development would be more suitable than its distributed development through modularised tasks. Despite their efforts to control overheads related to inter-team dependencies, additional overheads are introduced by their approach. The company may not be correctly handling highly-skilled employees at their lower-cost sites, leading to high levels of attrition.
Chapter 6. Cross-Case Analysis

6.1 Introduction

The previous chapter presented the empirical findings of the cases of this study in isolation. The empirical findings were followed by within-case analysis which highlighted the themes emerging from each case, and how the GSD benefits may be affected by those issues. In this chapter, the empirical findings from the cases are analysed on a cross-case basis to help address the research questions that drove this study. The aim of the cross-case analysis here is to deepen the understanding and explanation of the cases (Miles and Huberman 1994). While each case is unique and must be understood on its own terms, a comparative analysis of the cases can aid in understanding the cases and to answer the research questions. From that, the process can improve our understanding of realisation of the potential benefits of GSD. This chapter begins with an initial comparison of the two cases following on from the within-case analysis, and this is followed by a cross-case examination of each potential benefit.

6.2 Initial contrasts

While each case of this study stands in its own right, the approach of studying both cases was similar in its implementation. This allows for the cases to be compared, contrasted and analysed in relation to each other, while not ignoring the intricacies of each case. A better understanding of the differences and similarities of the cases helps to address why there are differences in the realisation of the GSD benefits. The within-case analyses of the previous chapter both led to similar emerging themes. Together these three themes were: 1) site hierarchy, 2) team structure, and 3) costing of projects (and overheads). The analysis attempted to identify which benefits are affected by these three emerging themes.

The emerging themes offer us increased understanding of the knowledge presented in the form of the framework of Chapter 2. These themes that have emerged from the within-case analysis deal with project-level organisational characteristics. Both cases have shown that there is an interplay between these characteristics and the potential benefits of GSD
(and the distances and processes upon which they are based). Indeed, for both companies
the primary and fundamental motivator to globalise software development has been to
reduce costs. This drive to reduce costs led to teams being set up in multiple locations across
the globe. However, the requirement to constantly communicate remained, thus exposing
the processes of communication, coordination and control to the effects of geographical,
temporal, and socio-cultural distance.

The findings from both cases both show that geographical and temporal distance in
particular affect the ability to communicate, and thus to coordinate and to control. In
Pennysoft, Irish-based managers expressed frustration with the overheads of having to
conduct such constrained communication. Semicon interviewees referred to the period
several years before the interviews took place, when teams from eight different locations
were attempting to coordinate their software development activities with difficulty. A
similar type of overhead manifested itself in both companies.

However, the process evolved further for Semicon, as it was felt that the overhead of
coordinating eight software development sites was taking too much effort. Semicon moved
away from the highly distributed project structure by enforcing two overarching rules: 1) not
more than two geographies would be involved in any one project, 2) teams would be
co-located and act independently. These objectives require Semicon projects to invest
effort into planning for independent components of the systems so as to be able to assign
to separate teams. As the within-case analysis of Semicon identified, this effort at the
beginning and end of projects leads to an overhead in itself.

For both cases, the researcher has argued that overheads being experienced in their GSD
activities are ultimately affecting the initial motivator of their GSD activities, that is to
reduce costs. Indeed, it is the motivation to reduce costs that has resulted in organisational
structures that are sub-optimal for the development teams. Figure 6-1 illustrates this
interplay between the objective of reducing costs with the effect it has had on the software
development process.
The process illustrated in Figure 6-1 shows that both companies first tended towards simply creating teams across global distances as required. The motivation to reduce costs by employing software development team members in lower-cost locations had a direct effect on the software development process in both companies. Pennysoft are still in this scenario, with managers attempting to control software developers who must coordinate daily with remote colleagues. As previously described, Semicon were motivated to minimise the overheads with such an arrangement. Semicon evolved further than Pennysoft by learning to avoid the costly coordination overhead, but in doing so introduced overheads of a different nature. This initial cross-case analysis highlights the findings of both case
studies – that it is not clear whether the overheads of pursuing reduced cost savings through GSD is worthwhile.

The ultimate purpose of this chapter’s analysis is to help describe how the interplay illustrated in the previous figure affects the companies’ realisation of all of the identified benefits. Through analysing the two cases against each other, this chapter attempts to understand why each particular benefit was or was not realised in each company. That analysis is presented in the following section.

6.3 A comparative analysis of the case findings

Both companies have differing approaches as to how to leverage globally dispersed personnel, and the set of benefits being realised by each company is somewhat different. In this section of the chapter, the benefits are compared and contrasted in light of both cases. To begin this analysis, Table 6-1 provides a summary of within-case findings side-by-side as to the realisation of each benefit. The researcher presented an early version of these findings at the International Conference on Global Software Engineering 2006 (Ó Conchúir et al. 2006).
Table 6-1. Summary of within-case findings from both cases.

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Realisation in Pennysoft</th>
<th>Realisation in Semicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proximity to market and customer</td>
<td>No, yet some proximity maintained for market and customers.</td>
<td>Yes.</td>
</tr>
<tr>
<td>Improved focus on documentation</td>
<td>No, not perceived as beneficial.</td>
<td>No, not accepted as a benefit of GSD.</td>
</tr>
<tr>
<td>Reliance on asynchronous communication</td>
<td>No, not perceived as beneficial.</td>
<td>No.</td>
</tr>
<tr>
<td>Access to large skilled labour pool</td>
<td>Yes. Secondary benefit to seeking out cheaper labour costs.</td>
<td>Yes, but perhaps not maximising their access to genius employees.</td>
</tr>
<tr>
<td>Access to cost-effective labour pool</td>
<td>Yes, but with considerable overheads.</td>
<td>Yes, but much effort to break cross-team dependencies for GSD.</td>
</tr>
<tr>
<td>Allocation of roles and team structure</td>
<td>Yes, thanks to flexible team structure.</td>
<td>Yes, at project rather than team level.</td>
</tr>
<tr>
<td>Cross-site modularisation of work</td>
<td>No.</td>
<td>Yes, with significant effort.</td>
</tr>
<tr>
<td>Improved process definition</td>
<td>No, not perceived as beneficial.</td>
<td>No, not perceived as a benefit of GSD.</td>
</tr>
<tr>
<td>Time zone effectiveness</td>
<td>Yes for certain tasks, but not a widespread benefit.</td>
<td>No.</td>
</tr>
<tr>
<td>Time zone efficiency</td>
<td>Yes in India, not in Ireland. Unknown for US.</td>
<td>No.</td>
</tr>
<tr>
<td>Innovation and shared best practice</td>
<td>No.</td>
<td>No particular evidence of this benefit in practice.</td>
</tr>
<tr>
<td>Increased autonomy</td>
<td>No.</td>
<td>Yes.</td>
</tr>
</tbody>
</table>

Table 6-1 shows that a minority of the benefits are being realised in each of the companies in line with the proposed framework. While there are differences in which benefits are being realised and how they are being realised, there are also similarities in the perception of some benefits. The following cross-case analysis examines these similarities and differences. The aim of this analysis is to provide a deeper understanding of the nature of each benefit by examining why the benefit is (or is not) being realised. With this additional understanding, we can approach the topic of how to optimise the realisation of the benefits.
6.3.1 Proximity to market and customer

Both case study companies benefit in some way from GSD in that it allows them to locate software development close to their customers. For Pennysoft, GSD has been leveraged to move development away from its traditional customers, while maintaining some presence on the US East Coast. Semicon has made use of GSD to stretch its software development towards its target markets by establishing permanent teams in the Far East. In practicality, Pennysoft is not realising this proposed benefit as according to the framework, but is experiencing a related flexibility. Semicon is indeed realising this benefit. The implication of realising this benefit is potentially stretching software development activities across geographical, temporal and socio-cultural distances.

In relation to this benefit, it was noted in both cases that external customers ultimately are not aware of the GSD activities of the companies. Rather, it is senior business analysts who deal directly with the customer to maintain corporate relationships and to gather business requirements for the software to be built. Customers are interested in the deliverables of a project, rather than how it is being implemented internally.

An analysis of the Pennysoft case shows that by stretching development into cheaper employment markets, it has directly resulted in extra effort for Irish-based managers. They must communicate directly with US-based business analyst during Ireland’s afternoons, and also coordinate with their Indian-based team members. However, the daily cross-site effort is not as obvious in Semicon, as the gathering of business requirements is managed more centrally by the core project team. Specifically, Interviewee H in Semicon noted that this benefit was not really visible to the software teams, as it is more the marketing team that deals directly with customers. Business analysts are closer to the customers, rather than much of the actual software development. Pennysoft’s Irish-based managers participate in more constant communication with client-facing analyst, perhaps due to a less centralised project structure. Although the case study data does not provide a full explanation for this, it does show that Pennysoft’s management are vulnerable to problems in everyday work tasks involving US-based personnel.
6.3.2 Improved focus on documentation & improved process definition

Two benefits (improved focus on documentation and improved process definition) are combined here in this sub-section as they were classified similarly in both companies. Within each company, the same conclusion has been identified for both potential benefits.

For both of these potential benefits, it was found in Pennysoft that the benefit is being realised, but is being perceived as an overhead rather than as a benefit. In Semicon, both benefits are being realised, but are not attributed to the characteristics of GSD. In Pennysoft, where the team structure implies on-going coordination between sites, interviewees felt that maintaining common documentation was an overhead in keeping it updated. However, it is certainly a necessity for the cooperation of their sub teams. Likewise, Semicon promotes maintaining good documentation. The up-front planning of modularisation of their systems ensures the creation of common documentation before any development starts. It is important for Semicon teams to follow the documentation, as their components are not fully integrated until the end of the project. As a conclusion, Pennysoft requires on-going updates to their documentation, even more so in the context of GSD, but the resultant documentation is not seen as a benefit. Documentation is also used to accommodate the quick re-allocation of team members, but is inadequate for this purpose. In Semicon, documentation is also a requirement for the independent teams, but this was not perceived by interviewees to be a result of GSD but rather the company’s engineering ethos. Perhaps GSD affects the point at which documentation is created in Semicon projects, pushing its creation to the planning phase only. The benefit of improved focus on documentation manifests itself in Pennysoft in order to complete on-going tasks within the distributed team and in Semicon for inter-team documentation.

The same perceptions were attributed to the potential benefit of improved process definition. In Pennysoft, it is acknowledged that processes have been improved in order to deal with the global distances, but that the increased focus on processes is not beneficial overall. Semicon employees see themselves as being strong in creating solid processes, regardless of the distances of GSD being involved. The within-case findings show us that Semicon are certainly more process-mature, and are more open to putting in place strong stable processes between all teams.
6.3.3 Record kept of communications

This potential benefit is more relevant to Pennysoft than to Semicon because of Pennysoft’s need to communicate daily across temporal distance. It is the team structure of Pennysoft that leaves their daily work exposed to temporal distance, and so timely asynchronous communication is more important to them. Both companies certainly depend on asynchronous communication, although Pennysoft focuses more on informal e-mail communications while Semicon focuses more on formal asynchronous communication.

This benefit was put to interviewees in both Pennysoft and Semicon, and no interviewee perceived this characteristic of asynchronous communication to be particularly beneficial. Instant messaging (IM), generally a synchronous communication method, was in use in both companies. As highlighted in the findings for Pennysoft, the benefit of having a record of conversation was countered with ability to communicate informally (and quickly) through IM due to no logs being kept. This finding itself is questionable, as IM clients have the ability to record conversations, although the researcher did not consider to follow up on this aspect during interviews. With respect to asynchronous communication specifically, there is a need in Pennysoft in particular to communicate across temporal distance. In face of this need, the findings have shown a strong organisational preference towards accommodating synchronous communication. E-mail as a communication medium was deemed to be limited, and interviewees expressed that it is not easy to convey nuances of opinion:

*E-mail is the common way of causing a dispute! It’s not easy to reach humour.*

*Telephone is better.* (Interviewee F, Pennysoft)

E-mail *does* indeed maintain a record of the communication, but as with improved focus on documentation, the benefit does not offer an ultimate benefit over its alternative (synchronous communication).

An analysis of this potential benefit at Pennysoft raises the broader issue of overcoming temporal distance. The findings have shown that employees at all three Pennysoft sites are under pressure to shift their working hours in order to maximise temporal overlap, thus reducing temporal distance. This demonstrates the importance of synchronous communication within software development teams.
6.3.4 Access to large skilled labour pool

From the findings of both cases, we see that this benefit is indeed attractive to the companies, but it is certainly secondary to the benefit of cost savings. It was more apparent in Semicon that the company seeks out highly skilled labour to incorporate into their GSD activities. Pennysoft have leveraged the skilled labour pool in Ireland to set up two development locations there (one being the case study site), and has now moved on to employ thousands of skilled employees in India. Semicon have access to “cream of the crop” employees in countries such as India, Malaysia and China with entire development teams there.

In Pennysoft it was highlighted that despite acceptable process maturity in India, the personnel there do not have sufficient business experience to be able to work independently in the global context. This has put a severe strain on the GSD teams in Pennysoft. Therefore, while Pennysoft does tap into the large employment market of skilled labour, those personnel add a burden to the existing teams. The ramp-up period of newly-established teams in India has spanned several years, and is on-going. During this time, sites in higher-cost locations have been dedicating resources to help train in the teams, adding to overhead in realising this benefit. This has included a buddy system where time is allocated daily to training in remote colleagues.

In Semicon, while attrition remains problematic, the benefits seem to be better realised than at Pennysoft. Semicon’s relative greater success may be due to the stability of their teams: all of their sites are already well-established and have reached a critical mass of skills and experience. This experience, in turn, has allowed the teams to act as peers within global projects rather than sit within a hierarchical chain as is the case in Pennysoft. While there is competition between sites to get the most valuable work, it is not the case that teams are attempting to bypass each other.

The findings from the two cases, have highlighted issues which challenge the realisation of this potential benefit:

- Attrition remains a huge concern for Pennysoft who employ in Bangalore and, even more recently, in Delhi. Concern of attrition was also raised in Semicon. When asked about the progress of this issue, Interviewee H in Pennysoft feared
that the rate of attrition would not reduce in the future, meaning that there may always be a problem with achieving sufficient business expertise in India.

- **Nature of work being sent offshore.** At Pennysoft, much Q.A. work is being sent to India, with more complex development assigned to the US and Ireland. Meanwhile, Semicon does not offshore the “crown jewels”, preferring to keep core business tasks in the US. These policies accentuate issues with high attrition rates, as employees may leave to find more interesting work. Both companies are risking the effectiveness of employing skilled labour offshore because of such policies.

- **Hierarchy.** The ambitions of the newly-established Pennysoft teams in India are at odds with the inter-site hierarchy in place. However, this hierarchy has arisen naturally due to the phased establishment of the Irish site, and later the Indian sites. The site hierarchy may be affecting perceptions of remote colleagues: Irish interviewees admitted that they may not fully appreciate the skills of their colleagues in India. Meanwhile, the Indian teams would prefer to work directly with the US, while they could be leveraging the skill sets and temporal overlap offered by Ireland. Such problems did not appear in the Semicon case because of the peer-based relationship of the teams.

- **Team maturity.** The lack of business maturity found in Pennysoft’s Indian teams means that they are not fully leveraging the resources there. It is also perceived that Indian personnel are less independent perhaps for cultural reasons, as they expect to be handed daily task lists. Frequent communication over instant messaging initiated by India is deemed unnecessary from the Irish side.

As a conclusion to this benefit, both companies are realising the benefit, but are not putting in place structures that maximise its effect on the organisation. As a counter-suggestion to the issues identified above, the researcher suggests the following structures to be favourable to realising this benefit, based on the findings and analysis of the two companies involved in the case study:

- **Nature of work being sent offshore.** Allow core corporate work to be distributed amongst competing sites. Maximise the leveraging of highly-skilled offshore staff by having them work on complex problems, potentially improving morale and thus reducing the rate of attrition.
• **Hierarchy.** While maintaining competitive structures between different sites, aim to flatten the inter-site hierarchy. This may reduce frustration of sites currently lower in the hierarchy, and would allow different teams to focus on different core competencies.

• **Team maturity.** Train in newer teams with the explicit objective of making them independent for their day-to-day work, enabling them to take responsibility for the completion of entire portions of projects. This again may improve employee morale, potentially reducing the rate of attrition.

### 6.3.5 Access to cost-effective labour pool

This proposed benefit has emerged from the multiple-case study to be the most important and sought-after of benefits. Interviewees in both companies stated that cost savings or cost effectiveness was the main reason for their companies to develop software in a global setting. Both companies are American-based multi-nationals, whose software development would remain at company headquarters were it not for cost and resource interests. Not only are salaries relatively lower elsewhere, but the cost of real estate and other overheads such as the cost of putting infrastructure in place are also motivating factors in this potential benefit, according to the interviewees.

To answer the research question of whether or not this benefit is being realised, the up-front answer from the study’s findings is ‘yes’. Both companies are indeed employing personnel in lower-cost markets at salaries that are potentially several times smaller than those in the US. For Pennysoft, the Irish site charges 73 percent of the rate charged by the US site; the Indian site charges 32 percent of the US fee. Pennysoft’s drive towards offshoring is evidenced by their policy to employ 40 percent of personnel ‘offshore’. For Semicon, the project costs are broadly similar, with the Irish site costing 75 percent compared to the US cost, and Indian projects costing 25 percent of the US cost.

However, the findings have indicated that there are many challenges to the realisation of such out-right cost savings. The findings of the two cases highlighted that the extra effort required to develop software in a distributed fashion is generally not taken into account when upper-management is choosing where to develop the software. As expressed by the interviewees of the study, it is questionable whether or not the potential benefit of
increased cost effectiveness is being realised. The case study findings highlight an on-going tension between the organisation’s corporate policies, and their negative effect on the workings of individual teams and team members who are ‘on the ground’. This tension manifests itself differently in both organisations due to the structure of their teams. Interviewees in Pennysoft expressed frustration at their daily working lives due to the constant overhead of communication across different time zones. This frustration is evidence that the efficiency of the team’s workings is far from optimal. Such frustration of daily work was not expressed by interviewees in Semicon. Instead it was found that their teams must cooperate intensely both when planning the system to be developed and when components of the system are being integrated. In Semicon, while teams offer various levels of cost advantage, they must still spend much effort in coordinating with other teams which includes travel of key team members.

The question of the realisation of this benefit brings us back to Figure 6-1. In the process of attempting to achieve their goal of reducing costs, both companies experience overheads as part of their approach to GSD. Pennysoft have significant overheads from on-going cross-site communications, whereas Semicon have significant overheads in attempting to reduce that same cross-site communication. Therefore, the findings of both cases bring into question to what extent a company may realise this potential benefit given their approach to GSD.

6.3.6 Cross-site modularisation of work

The case study companies differ significantly according to this proposed benefit. In Pennysoft, it was recognised that modularising work tasks to reduce cross-site dependencies was possibly advantageous, but was not even being enforced as an ideal to strive towards. Interviewees felt that this benefit could not be realised due to the relatively small teams which are distributed across several locations. Generally, it was felt the software development activities at hand were too complex to be divided into well-defined portions of work. However, other specific tasks such as system configuration and testing are more suited to modularisation, as portions of the work are smaller and can be assigned to a single individual. It should also be noted that Pennysoft do not ignore the importance of how and where tasks are allocated:
In contrast, however, Semicon’s approach actively pursues the modularisation of work tasks so that a defined portion of work can be developed locally, independent of remote expertise. This approach is at odds with Pennysoft’s claim that the nature of their work is too complex to modularise. Semicon’s stated aim is to have “loosely-coupled and highly cohesive teams” – the co-located teams should not be heavily reliant on other teams’ work, and the work being completed by each team should be tightly integrated. Semicon’s motivation for this structure stems from their previous experience of being overly-exposed due to global distances being prevalent within their software projects. With an early focus on system design, Semicon attempts to define cohesive system components that can be developed in relative independence.

Semicon’s approach is not perfect. There is regular planned communication that must take place between teams. There is significant effort at planning, design and integration phases. As previously noted, there is a difficulty with optimally re-using code and reducing dependencies on remote expertise.

There is a significant difference in size and time scale of the work tasks being allocated at Pennysoft and Semicon. Pennysoft currently view task allocation as a daily activity for individual developers, as illustrated in Figure 6-2. This figure shows that at Pennysoft, the interviewees’ perception of task allocation was very much at an individual level. This takes place at a daily level, and often across different sites. Interviewees argued that the nature of their software development doesn’t allow them to give developers a large enough piece of work to make the independent over a period of weeks or months. Given the high degree of interdependence between developers at all sites, it was not seen that a sub-grouping of developers at one location could perform their work alone over an extended period of time.
In contrast, Semicon’s view is significantly more long term. As illustrated in Figure 6-3, task allocation is viewed at a team level, where such tasks are entire system components to be developed over a period of a number of months. While of course there is still task allocation taking place at a daily level, Semicon teams have managed to abstract modularisation of work to view it at a level of teams and system components. In Pennysoft, perhaps by forming sub-teams of co-located team members, for example, they could begin to approach modularisation of work at a sub-team rather than individual level.

Apart from level of task allocation discussed above, it seems that the realisation of this benefit can have an impact of other benefits, through the communication pathways it enforces. Pennysoft fail to realise this benefit and, consequently, their team members rely
heavily on coordination with remote colleagues. Rather than reducing cross-site dependencies, Pennysoft teams operate at full exposure to geographical and temporal distance. Interviewees complained of the difficulties related to coordinating with colleagues in several other locations. It is difficult to manage remote team members. Delays can arise because of the time it can take for a remote colleague to reply. Team members are required to work non-standard working hours. This relates to the previous configuration of teams in Semicon where teams from eight separate sites were attempting to coordinate to build software. By properly modularising work tasks to reduce cross-site dependencies, Pennysoft could more efficiently leverage the cheaper offshore labour towards which they have been striving.

The lack of task modularisation at Pennysoft may also be contributing to tensions that have arisen from site hierarchy. Pennysoft interviewees recognised that giving ownership to the Indian sites for portions of the work was necessary to manage the tensions between them:

   Our strategy is to give everyone a share of the ownership. Therefore everyone has a stake in the project, and as such, the project works well. (Interviewee F, Pennysoft)

However, the strategy described by Interviewee F was expressed more as an objective rather than the current practices at the case site. As already described, ownership of project firmly rested in the US, but with Ireland taking responsibility for the technical completion of the projects. At the same time, Indian personnel in Pennysoft are currently unable to act independently of the Irish site because of the heavy cross-site dependencies. By modularising work tasks to reduce cross-site dependencies, the Indian sites could begin to work more independently of the Irish site, thus reducing the current tensions.

Semicon aim to realise this benefit, while it is difficult to achieve the benefit effectively. The modularisation of work tasks allows Semicon to reduce the effect of the global distances that divide teams, by allowing non-dispersed teams to operate relatively independently of each other. Semicon’s modularised approach also allows for the phased release of system modules to customers. Semicon view this approach as advantageous compared of the coordination complexities that they had experienced in the past. While this approach reduces daily coordination complexities, it also conflicts with the proposed benefits of time zone effectiveness (due to the lack of daily coordination across time zones) and allocation of roles and team structures (due to the fact that a team is not allowed to incorporate a remote team member).
As a conclusion, the realisation of this benefit interacts closely with the benefit of reduced costs, as implementing the benefit implies significant allocation of resources to manage the effort. When deciding on the best approach to modularisation of work, the benefits and challenges to both approaches as laid out above should be taken into account. In the researcher’s mind, Semicon’s approach, while generally less flexible, is more sustainable due to its focus on reducing costly daily interdependencies between sites. When this finding was presented to Pennysoft, the managers still concluded that the nature of work involved, plus the lack of business maturity at the Indian site would prohibit them from better realising this benefit. Perhaps as the Indian site does approach maturity, the company could take a different approach.

6.3.7 Time zone effectiveness

This benefit was mainly not being realised in the Pennysoft case study site in Ireland, but may have some relevancy to the Indian site with their morning working hours not overlapping with other sites. The benefit is not relevant to Semicon. The difference between the ability to realise this benefit is based upon the team structure – it is only relevant where there is on-going coordination between teams in different time zones.

The within-case analysis of Pennysoft showed that the benefit is only being realised for a limited number of tasks. Overall, however, team members are under pressure to work non-standard hours to minimise the effect of temporal distance, thus limiting the positive effect of this benefit on the organisation. Pennysoft personnel rely on remote colleagues for their day-to-day development work, and those colleagues can be five to eight time zones away. While several Semicon interviewees mentioned the need to work non-standard hours, the Irish Pennysoft interviewees complained openly of the strains of working with colleagues in other time zones. Indeed, Semicon interviewees agreed that the benefit of follow-the-sun development is irrelevant for their teams, as developers generally do not deal with remote colleagues on a day-to-day basis.

Interestingly in Pennysoft, Interviewees E and H noted problems of working with US-based colleagues who were not prepared to be more flexible with their working hours. At the other end of the spectrum, Indian-based teams tend to work very late in order to increase temporal overlap with the Irish and US teams. Indeed, Indian-based Interviewee D was working in a team which works in two shifts. The first shift works standard hours,
and the second shift works late into the evening to coordinate with other sites. The interviewee noted that Indian-based personnel were prepared to work like this as they understood the requirements of working for multi-national companies. India’s near-subordinate role in the organisational structure reflects the extent to which they work non-standard hours.

In summary, with respect to the benefit of time zone effectiveness, it is only being realised to a limited extent in Pennysoft. While Pennysoft’s team structure is well-suited to time zone effectiveness, team members actually work non-standard hours in order to overcome the time zones that divide them. The proposed benefit is irrelevant to Semicon’s overall approach as it would require close interdependences between sites, which Semicon does not allow for.

The analysis of this and previous potential benefits has brought to light an important aspect of the opposing approaches of the two case study companies, and the effect each approach has on how benefits are being realised. The case study companies attempt to localise GSD to both avoid certain problems, and to strive towards realising certain benefits. Pennysoft’s approach ‘localises’ GSD by tending to eliminate the temporal distance between people. As such, team members shift their working hours in order to maximise temporal overlap with remote colleagues. As previously quoted, Interviewee G in Pennysoft said the working day of Ireland is “not that far off” the working day in the US because of the shifting working hours at both locations. In contrast, Semicon localise their GSD activities by minimising the effect of geographical distance by forcing all teams to be entirely located at one site only. The minimisation of geographical distance also allows for reducing temporal and socio-cultural distance within teams. While both companies approach the complex problem of GSD differently, both attempt to simplify the problem by reducing effect of the global distances.

6.3.8 Time zone efficiency

While time zone effectiveness deals with maximising the number of hours during which software is being developed, time zone efficiency refers to the potential increase in efficiency given that temporal distance reduces the level of on-going synchronous communication. The findings showed that this benefit was only being realised by the Indian site of Pennysoft, and is again irrelevant to Semicon due to their co-located team
structure. Indian-based Interviewee C at Pennysoft agreed that a positive effect of time zones meant that cross-site meetings only occurred during a specific portion of the working day. However, with the study’s focus on Irish-based teams, neither case had evidence of realisation of this benefit in Ireland. Indeed, the broader findings from Pennysoft suggest that this benefit is far from realised. Rather than temporal distance between colleagues being perceived as helpful, it was blamed for increasing coordination complexity and the related problems that have already been discussed. The Pennysoft findings show a strong preference for synchronous communication, over any potentially beneficial aspect of asynchronous communication. For Semicon, the benefit is made irrelevant by the fact that teams aim for minimal levels of cross-site coordination, and so there is no benefit to be gained across time zones.

6.3.9 Innovation and shared best practices

Neither case study company deemed this potential benefit to be important. As such, relatively little data in the study’s findings relate directly to this benefit. The findings of the Pennysoft case show that while several interviewees acknowledge that this may be beneficial for their teams, the spreading of best practices is promoted in a top-down fashion from upper management. Interviewees were sceptical whether the benefit is real, and indeed Interviewee I argued that the levels of innovation were not any higher than they would be in a purely co-located setting. Perhaps site hierarchy is negatively affecting the realisation of this benefit. The US site rejected proposed processes that were developed by Ireland and India. Similarly, the Irish site may be rejecting best practices that are being suggested by India because of a lack of respect of their abilities.

Semicon interviewees did not perceive this benefit to be realised. This could perhaps be down to the fact that Semicon teams are in single locations, and that daily contact between teams are kept to a minimum, limiting the natural flow of best practices. The ‘localised’ development at Semicon may be limiting the globalised benefit of innovation arising from different national backgrounds. Perhaps the newly-enforced policy of code re-use will help the spread of innovative software development between teams.
6.3.10 Allocation of roles and team structure

Both case study companies realise organisational flexibility, albeit at different levels in their organisational structures. Pennysoft realise the benefit at an individual level by reassigning people to different teams when the need arises. For example, if several developers end their work with one project, they may join another existing project with distributed teams. Pennysoft’s approach to a distributed team structure allows it to leverage a skilled employee no matter where they may be (nevertheless, the ramp-up costs must be taken into account).

For Semicon, this within-team flexibility is not realised because their teams are only comprised of co-located team members. However, an entire team can be added to a project when the existing capacity of the current teams is not sufficient. This would only be possible if that team was within the two “geographies” limit imposed by organisational policies. Semicon’s project structure is flexible as to whether the additional team is in-house, or added through offshore outsourcing.

Overall, this benefit is being realised by both companies over an extended timeframe. The proposed up-front simplicity of re-allocating resources was rejected in Pennysoft, as it takes time for new personnel to familiarise themselves with the new project. Also, adding new resources to a project does not imply that existing personnel can be instantly freed up, as they must also dedicate time towards training in their new colleagues. As a conclusion, the realisation of this benefit within teams exposes the team to additional overheads. Whether in Pennysoft of Semicon, a re-allocation requires ramp-up time, and so is not cost-free.

6.3.11 Increased autonomy for software development units

The Pennysoft findings have shown that the opposite of this potential benefit is in fact the case there. Geographical distance within the Pennysoft teams does not imply increased autonomy, as team boundaries are not defined by geographical distance. As such, the Indian personnel have not gained autonomy from the Irish site as a result of their geographical distance. This will probably improve as India develops its expertise over time. In contrast, Semicon teams are given the mandate to work independently of other teams, therefore implying a high degree of autonomy for each team. The benefits of reducing cross-site dependencies are valuable for Semicon, and therefore this benefit is of
importance to them. In both cases, it is the team structure that defines the realisation of this benefit.

6.4 The realisation of the benefits of GSD

The cross-case analysis presented in this chapter has compared and contrasted the approaches of Pennysoft and Semicon, and how the realisation of the benefits varied according to the different approaches. The cross-case analysis has allowed us to show the similarities between the cases, illustrated by Figure 6-1. Given the team structure (and less so, the site hierarchy), we have seen different effects of geographical, temporal and socio-cultural distance on the realisation of the benefits. Table 6-2 below provides a simplified summary of the realisation of the benefits of GSD, whether they are at least partially realised by Pennysoft only, Semicon only, by both, or by neither.

Table 6-2. A distribution table of the realisation of the benefits of GSD.

<table>
<thead>
<tr>
<th>✓ Pennysoft only</th>
<th>✓ Semicon only</th>
<th>✓ Both</th>
<th>✗ Neither</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time zone efficacy</td>
<td>Proximity to market and customer</td>
<td>Access to large skilled labour pool</td>
<td>Improved focus on documentation</td>
</tr>
<tr>
<td>Time zone efficiency</td>
<td>Cross-site modularisation of work</td>
<td>Access to cost-effective labour pool</td>
<td>Record kept of asynchronous communication</td>
</tr>
<tr>
<td></td>
<td>Increased autonomy</td>
<td>Allocation of roles and team structure</td>
<td>Improved process definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Innovation and shared best practice</td>
</tr>
</tbody>
</table>

Table 6-2 offers an interesting insight into the realisation of the benefits according to the case study data. When we take a look at the benefits only realised by Pennysoft, both relate to temporal distance. The realisation of these benefits is possible for Pennysoft rather than Semicon because of the feature of temporal distance within their teams. The possibility of these benefits is not significant in Semicon because the effect of temporal distance is minimised within the teams’ environments.

Further, if we look at the benefits only realised to some extent by Semicon, we find that they are related to the independence of their teams, by reducing the effects of geographical distance (which in turn reduces temporal and cultural distances). Teams are able to act
relatively independently because of the modularisation of work tasks, and so colleagues work together in the same sites.

In neither company did participants agree that increased focus on documentation and processes were beneficial, or at least they were a necessity without any further benefit. The difficulty of maintaining cross-site documentation points us to the other proposed advantage of performing asynchronous communication, that is that a record is kept of the communication. Both companies’ findings show us that they are minimising the need for asynchronous communication altogether – Pennysoft are doing so by minimising temporal distance, and Semicon are doing so by minimising geographical distance.

6.5 Summary

The cross-case analysis presented in this chapter follows the advice of Miles of Huberman (1994), allowing us to gain deeper insights into the phenomenon. The initial comparison of the cases led to the identification of the interplay of the distances, processes and emerging themes, helping us to understand the similarities between the cases. The remainder of the chapter looked at each potential benefit, examining the findings coming from both cases. This was concluded by Table 6-2 which gives an overview of the distribution of the realisation of the benefits across the two cases. The following chapter builds upon the cross-case analysis of this chapter, and discusses the implications of the findings in relation to previous studies.
Chapter 7. Discussion

7.1 Introduction

The cross-case analysis of each potential benefit in Chapter 6 was presented in order to better understand the findings coming from each case in this study. The analysis described the interplay between the distances, processes and team structures in relation to the realisation of the benefits. The chapter presented in Table 6-2 a distribution table of which benefits are being realised through the two different approaches to GSD. This chapter discusses the implications of the two different approaches to structuring GSD teams, and how that affects the realisation of the benefits. The chapter first discusses how the companies involved are in effect localising their GSD activities in an attempt to reduce the negative effects of working across distance. Then, team configurations are discussed along with task allocation for these teams, as this implies at what stage team members must coordinate to complete a project. It is then argued that the corporate-wide benefits are being pursued at the expense of the software development practitioners at project and team level. Finally, the findings for each benefit are presented in the context of previous studies.

7.2 The ‘localisation’ of GSD

The findings of this study of the two case study companies, Pennysoft and Semicon, show that it is too simplistic to assume that all benefits can be automatically realised when all three distances of GSD are in full effect. Both companies have shied away from exposing themselves to the full effects of geographical, temporal and socio-cultural distances, even though these global distances form the very characteristics of the phenomenon. In effect, the companies are localising their globalised software development (the term adopted here is not meant to relate to the separate concept of localisation). This very action is in contradiction with an assumption that great benefit can be gained from situations that involve full exposure to the distances. While Carmel (2006) stated that temporal distance is inherently not solvable, Pennysoft are tending towards localising their teams by minimising temporal distance. This is achieved by all sites shifting and stretching their daily working hours. Large temporal distance increases the need to communicate asynchronously. The
ineffectiveness of asynchronous communication in a software development environment warrants the effort required to deviate from normal working hours. Semicon have in effect localised their GSD by choosing to keep all team members in one place, thus reducing the geographical distance between team members. This, along with reduced temporal and socio-cultural distance, allows team members to benefit from working face-to-face, without the expense of asynchronous communication.

7.2.1 Reducing temporal distance

Temporal distance itself is a characteristic of GSD that has been cited as the source of benefits of GSD, including follow-the-sun development (Carmel 1999). It does, however, bring with it documented issues for software development teams. Pennysoft experiences plenty of the related issues because temporal distance cannot be eliminated within the current team structure. The issues visible within Pennysoft have been previously documented, including excessive delays in awaiting feedback (Carmel and Agarwal 2001, Kiel 2003), and the requirement to shift working hours (Carmel 2006).

While Carmel is one of the main proponents of the benefits related to temporal distance, he has also acknowledged a finding similar to Pennysoft’s. He observed in Infosys in India (Carmel 2006) that even this organisation which specialised in “global delivery” must shift working hours to reduce temporal distance. Rather than aiming for follow-the-sun development, Carmel stated that the Infosys employees worked outside of standard working hours in order to facilitate collaboration with remote colleagues and clients. This suggests that one of the largest and best-established IT offshoring service providers in India favours temporal coverage, rather than attempting to leverage time zone effectiveness in their favour, similar to the approach by Pennysoft.

As Karolak (1998) pointed out, communication is less effective when delays are introduced due to temporal distance. Although Pennysoft was the only of the two case study companies to somewhat realise time zone-related benefits, team members are still forced to reduce temporal distance to improve their ability to communicate. By improving communication, the processes of coordination and control are also improved.
7.2.2 Reducing geographical distance

Carmel and Abbott (2007) have argued that despite reducing temporal distance between sites, geographical distance still very much matters. Semicon have had previous bad experience with software projects where the fundamental processes were fully exposed to the distances involved. Rather than reducing temporal distance itself, Semicon have localised their software development activities by having teams of co-located members. In effect, Semicon have bubbles of local software development taking place within the overall global projects. Geographical distance has been minimised between team members, which also implies that they practice every-day face-to-face meetings with minimal temporal distance. Chapter 2 has covered the importance of face-to-face communication in software development. In essence, it allows for a richer medium for the process of communication, allowing for misunderstandings to be quickly ironed out (Curtis et al. 1988). This communication richness is also important in helping to maintain a sense of teamness amongst colleagues (Carmel 1999).

Although teams in Semicon are localised, the distances between the different teams continues to expose projects to related problems. For example, key team members must still travel across great geographical distance to meet face-to-face for planning and integration. Inter-team communication during the software development process does indeed continue. Even then, Semicon are exposed to “big bang” integration, as warned by Battin et al. (2001). Grinter et al. (1999) reported of teams being “constantly surprised” of decisions being taken at other sites, and this is amplified by the relative lack of communication between sites. More recent studies focusing on knowledge sharing (Kotlarsky and Oshri 2005, Kotlarsky et al. 2008b) offer findings that seem to reject Semicon’s approach due to knowledge-related issues such as difficult integration. Those studies emphasise the need to share knowledge between sites rather than attempting to break communication between them. In light of these studies, a less extreme approach by Semicon may indeed improve their software development activities overall.

7.2.3 Realising the benefits in localised contexts

The findings of the Semicon case showed that leveraging temporal distance is not of interest to them, as they regard the software development process as months-long rather than an activity that should require daily cross-site interaction. Indeed, Semicon’s
localisation of GSD through pockets of localised development does affect the possible benefits they can realise. On the one hand, they can realise benefits relating to modularisation of system and tasks, and relative autonomy of teams. However, on the other hand the minimisation of temporal distance between colleagues has lead the teams away from considering practicing related benefits such as time zone effectiveness and time zone efficiency. As described, interviewees expressed that these benefits simply had no place in their approach to software development.

Pennysoft are more open to these temporal-related benefits despite attempting to reduce temporal distance. The lack of location-centric teams in Pennysoft does mean that they cannot easily aim for highly modularised tasks, and their sites are far from realising full autonomy from each other.

The concept of localising GSD activities brings us to a conclusion that the inherent global distances of the phenomenon are not the primary source of related benefits. While temporal distance may still offer advantages, for example, the effects of the distance are not the motivating factors in the companies practicing GSD. Rather, global distances are introduced as a side effect of the primary driving benefit of reduced development costs, along with the benefits of access to large skilled labour pools and proximity to market and customers. With a focus directly on the realisation of potential benefits of GSD, the findings of this study have shown that distance-related benefits are still secondary to the realisation of business-driven benefits of reducing costs while accessing abundant resources. Moreover, Carmel and Abbott (2007), after examining the phenomenon of ‘nearshoring’, conclude that the effects of distance on offshoring will continue to persist in software development into the foreseeable future. This implies that the tension of ‘localising’ instances of software development may continue as has been seen in the case study companies.

The companies are driven to localise aspects of their distributed software development, rather than attempt to leverage the potential benefits of truly globalised software development teams. Thus, the findings have shown that there is a tension between benefits sought after by the corporation and the effect that the realisation of these benefits has on the software development process. This tension is further discussed later in this chapter.
7.2.4 Discussion

While this multiple-case study has offered one perspective on realising the benefits in a ‘localised’ setting, others have been studying ways in which to reduce the actual distance. The study by Carmel and Abbott (2007) mentioned in the previous section looks at the phenomenon of ‘nearshoring’. They describe the option of finding a compromise between very cheap offshoring in a far-off nation compared with locating development in a country that is closer in terms of time zones, and perhaps in terms of culture. A related compromise is implementing a ‘bridging’ model where a country is used as an intermediate between two sites that are far apart (for example, Ireland acting as the bridge between the US and India). This phenomenon was reported separately based on the multiple-case study described in this thesis (Holmström Olsson et al. 2008). Milewski et al. (2008) take the view that such bridging techniques are of benefit since they introduce shorter communication ‘paths’ between sites. Their study includes the consideration of creating bridges between compatible cultures (in effect extending the localising notion of nearshoring). Studies focusing on the benefits of GSD while theorising on such bridging models may reveal further insights into the realisation of the benefits of GSD.

7.3 Team characteristics for realising benefits

The findings of this multiple case study indicate that the structure, size and maturity of teams may have an effect on which benefits may be realised. Not only have we seen that characteristics of Pennysoft and Semicon teams facilitate different benefits, an inefficient GSD team structure may indeed cancel any of the benefits being pursued. The following sections discuss the effects of team configuration, specifically team distribution, size and maturity.

7.3.1 Team distribution

Ebert et al. (2001) showed that small projects in Alcatel with highly scattered resources showed less than half the productivity compared to projects with fulltime staff. When comparing the opinions expressed by Pennysoft and Semicon interviewees, it seems to indicate that the Pennysoft teams are finding it relatively more difficult to cope with daily work, given the distribution of their teams. For this study, distribution was defined to cover
physical distribution across geographical and temporal distance, along with socio-cultural distance.

Previous publications have studied the various levels of team and project member distribution possible in GSD. Just as Pennysoft and Semicon follow different team structures, Kobitzsch et al. (2001) stated that organisations may choose to have multiple teams across multiple locations, or one team spread across multiple locations. Gumm (2006) set out to characterise her case study project by the distances involved, and concluded that organisational distance can be as problematic as physical distance. Evaristo and Scudder (2000) identified ten aspects of distribution within teams varying from the “need for synchronicity” to the types of stakeholders. Prikladnicki et al. (2003a) looked at the different team distributions such as in different locations of the same city, compared with more global distribution. Further to those studies, O’Leary and Cummings (2007) attempted to develop a model to measure the level of distribution for different configurations of distributed development teams. Dubé and Paré (2001) defined highly globalised teams (such as those found at Pennysoft) to be “global virtual teams”. Dubé and Paré found that in such teams, compared to “more localised virtual teams”, team effectiveness can be affected by cultural distance, communication problems, and discrepancies in technical proficiency. Indeed, the Pennysoft case findings showed in particular the overheads involved with having to communicate across global distances, and that there were mismatches in the level of technical experience expected of offshore resources.

Semicon’s co-located teams are less extreme examples of GSD teams, although their projects remain inter-continental. Ebert and de Neve (2001) argue strongly in favour of coherent, co-located teams with fulltime team members. They argue this based on the findings that such teams were twice as efficient as completing their tasks. This study, however, has found that such teams are less open to the potential benefits of the global distances even if such benefits such as improved innovation and shared best practice were not found to be realised at Pennysoft.

7.3.2 Team size

Pennysoft interviewees argued that their teams are too small to be able to break them into independently acting groups of developers. The reported size of teams was between
approximately 10 and 30 team members. In Semicon, it was reported that entire projects typically involved 50 people, however the size of projects in the case study varied between 9 and 40. The team sizes at Semicon in practice are not greatly different to those at Pennysoft. Therefore, the conclusion of the Pennysoft interviewees must be questioned. The ability of different team sizes to allocate different software development tasks is beyond the data of this case study. Lamersdorf et al. (2009) found that the allocation of tasks in GSD teams is success-critical, but they argue that there is an insufficient understanding of the criteria involved in such allocation. Some of the criteria include the availability of developers at particular sites and their level of expertise. Perhaps a workable team size will simply require a match between the type of work involved, and the level of skill and expertise of the developers.

7.3.3 Team maturity

The survey findings reported on by Global Insight (USA) Inc. (2005) found that the benefit of cost saving does depend on the maturity of the organisation, along with the type of activities involved. From the findings of this multiple case study, there is a greater differential in team maturity in Pennysoft compared to the teams of Semicon. The differential in expertise caused by the gradual progression towards GSD, and a site hierarchy naturally arose from this imbalance. Viewing software development as knowledge work, experienced-based knowledge cannot be replaced formulaically by models, processes, and standards (Steen 2007). It is not necessarily easy for teams to develop and mature given the difficult nature of GSD. In any context, a software developers must grapple with incomplete information coming from their colleagues and tools, only to be made more difficult in a distributed setting (Rönkkö 2007).

The effects of an imbalance of team maturity in Pennysoft have been discussed, and include the tension it creates between different sites. The relative immaturity of teams at Pennysoft was shown to limit the level of autonomy that can be given to a site. This relative lack of expertise at the Indian sites means that the Irish team must “hand hold” them on a daily basis. In effect, the lack of team maturity demonstrated at Pennysoft results in additional management overheads with dealing with resources that are based far away in other time zones. Levina and Vaast (2008) found that large disparities in salary levels accentuated the divide and imbalance between sites. Paradoxically, a future rise in maturity of teams in Pennysoft may promote a peer-based relationship between different sites,
although this would also imply increased costs in offshore locations, potentially eroding the attraction of practicing GSD.

The report by Hayduk (2003) concludes that cost savings of 25 percent can be achieved only after one to two years for long-term projects with at least 40 team members. At the time of the interviews at Pennysoft, India had already been involved for two years, but were not performing optimally. Prikladnicki et al. (2003b) agree strongly with such a finding, stating that GSD in itself is a maturity process. This is at odds with Carmel and Tija (2005) who argue that GSD projects can ramp-up to be operative within “days and weeks” rather than months and years. In contrast, the case study team at Semicon was fully functional when it was acquired by Semicon, and so there is less evidence of issues with the ramp-up of new sites. That being said, Semicon are establishing new teams in Asia, and so similar problems may result for them going forward.

7.4 Task allocation

The cross-case analysis of the previous chapter, and the previous sections of this chapter, have discussed how the realisation of the potential benefits of GSD are affected by the interplay between the distances and processes of GSD in two different approaches to team configuration. The findings of this multiple-case study place much emphasis on how the processes of communication, coordination and control are managed across geographical, temporal and cultural distances.

Team members must communicate whenever necessary to make the team efficient (DeMarco and Lister 1999). The two case study companies differ on when that communication is necessary. Communication is necessary between inter-dependent team members in order to manage those inter-dependencies. Thus, the process of coordination is vulnerable to communication problems, and these problems are magnified when having to communicate across distance. Task allocation plays a fundamental role in this mixture, as it can, in effect, define which resources will be dependent upon each other. Indeed, as is described in the following sections, the coordination pathways between team members affect the structure of the system they are developing.

There have been various suggestions on how to allocate tasks among GSD teams which are globally dispersed. Each method involves a balancing act between high task cohesiveness
and low task coupling. The implication is that if a team can be assigned to the development of a highly cohesive component with low coupling on other components, the team itself will be relatively independent from other teams. As was previously described in Chapter 3, an effective product architecture allows for GSD teams to work harmoniously without “stepping on each other’s toes” (Carmel 1999, p. 127). Since the resultant task allocation can be used to reduce coordination overheads, product architecture is an important coordination mechanism (Herbsleb and Grinter 1999a), as Semicon have recognised. Again, it is from such coordination overheads from which Pennysoft teams suffer. The following sections describe some of the implementation strategies for task allocation in GSD.

7.4.1 Allocation based on product structure

GSD project teams can be defined based on the structure of the product being developed. For this approach, tasks are allocated according to cleanly-separated functional system parts according to the product structure (Carmel 1999). These defined components should be allocated for the entire lifecycle (Battin et al. 2001), only to be brought together at a final integration step.

At Semicon, systems are carefully designed to limit the coupling between modules, while maintaining high within-module cohesion. Then, modules are allocated to the software teams in the hope that the teams themselves can benefit from minimal dependence on remote expertise. There are benefits to this clean separation of work, such as allowing teams to follow their own processes and use a specialised toolset. This is acceptable as long as the end-product complies with the interface requirements defined during the initial design period (Grinter et al. 1999). This reduces the coordination and technical costs, and schedule risks (Karolak 1998). The Semicon case showed that the system specifications must not change after the planning stage so as not to cause scheduling issues for multiple project teams. The approach also allows Semicon to allocate a module to a team external to the organisation, as also argued by Karolak (1998).

While this approach works relatively well for Semicon, the study findings have shown that it also requires much effort at planning, design, and integration phases, as has also been found in previous studies (Grinter 1998; Battin et al. 2001; Herbsleb et al. 2005). Such a rigid team structure may lead to a redundancy of skills between sites, which should be
avoided (Ebert and De Neve 2001). For example, all sites may require quality assurance skills, although not all of those skills will be in used at all sites at all times.

7.4.2 Phase-based allocation

Work may be divided and allocated by development life-cycle phases (Karolak 1998, Carmel 1999). Different phases in the software life-cycle include requirements engineering, design, implementation, testing and customer support. For example, one site may design the system architecture, while another site carries out the actual software development, while another carries out testing. Certain sites may have different competencies than others, especially if they have been acquired as part of an acquisition of another organisation (Carmel 1999, p. 134, Grinter et al. 1999, Ebert and De Neve 2001). The advantage of this is that teams can focus their skill set and tool set for carrying out specific tasks related to certain life-cycle activities (Karolak 1998). In fact, Karolak points out that outsourcing vendors specialised in certain life-cycle phases may be used when outsourcing phases of the project. He also states that phase-based allocation is most cost-effective in large projects.

According to Carmel, this approach uses the classic “throw-over-the-wall” paradigm. A team receives specifications, produces the required work, and throws the work “over-the-wall” to the next team in the process, while the first team moves on to their next assignment. Allocation strategies may become phase-based if cost is a driving issue (Carmel 1999). In particular, coding and testing tasks may be allocated to low-cost sites.

7.4.3 Virtual Teams

A virtual team comprises team members who are geographically dispersed and who may rarely meet, if at all, during the course of the project (Dubé and Paré 2001). The virtual team approach closely matches the model seen in Pennysoft, although the teams are comprised of bubbles of co-located team members. Virtual teams offer organisations a more flexible, agile, responsive, and inexpensive approach (Carmel 1999, p. 13). The findings of the Pennysoft case do suggest that this team structure allows for more flexible task allocation, however it is arguable that this approach is “inexpensive”. Virtual teams allow organisations to partner high-cost and highly-experienced engineers with lower-cost and less-experienced engineers located elsewhere (Casey and Richardson 2004), hence
facilitating the use of lower-cost labour. The findings at Pennysoft include additional factors to those reported by Casey and Richardson, as Pennysoft is experiencing high costs of training remote team members over several years.

Global virtual teams face particular problems with technological issues concerning compatibility and accessibility across sites, with cultural distance also posing problems (Dubé and Paré 2001). This is due to the fact that the team structure allows for developers to be heavily dependent on remote colleagues, forcing them to coordinate across geographical and temporal distances. Carmel (1999, p. 14) argues that this leads to a “networked organisation” of development sites, with communication pathways running between many of the sites due to interdependencies.

7.4.4 Points of weakness of each approach

Each of the strategies defined above have “points of weakness” at different times of the lifecycle (Carmel 1999). Those points of weakness are the intersections where project members must coordinate with others who are located at separate sites. Pennysoft’s points of weakness are daily, as each site in the virtual team arrangement is heavily dependent on others. Semicon’s daily work is less at risk to points of weakness, but they must exert much effort at the planning, design and integration phases of projects. The costs associated with each approach’s points of weakness should be taken into account when weighing the level of benefits such as cost savings that can be achieved. Phase-based allocation, not really seen in this multiple-case study, introduces a point of weakness at the end of each phase when there is a hand-over from one team to the next to begin work on a new phase. It is important for practitioners to understand how to measure the degree of team interdependence so that the effects of various levels of coupling can be understood and addressed (Herbsleb et al. 2000).

The advantages of each method of allocation may be eroded by the points of weakness that they incur. Organisations must decide on which allocation is best suited for their type of work and according to the dispersion of skills. There are good reasons for each approach (Carmel 1999, p. 135). Rothman (1998) argues that allocating teams by modules according to product structure is the best approach, as each team has all the expertise needed to complete their part of the work, and hence reducing their dependency on other teams. However, this method can lead to skills redundancy, and is not as flexible as virtual teams.
in allowing the organisation to make use of globally-dispersed resources. Notably, Herbsleb et al. (2000) reported that teams in their study naturally moved to greater site interdependence over time. Perhaps a similar outcome may arise in Pennysoft after a number of years. The phase-based approach may be best suited to projects with well-defined tasks which are easier to allocate by phase (Carmel 1999, p. 135). Otherwise, a module-based approach may be more suitable, where the initial system structure is more “abstract” and is therefore best to allocate to one team for the entire life cycle (Carmel 1999, p. 135).

7.5 Tension between corporate-related and other benefits

When discussing the localisation of GSD in a previous section, it was stated that core benefits of GSD being pursued by the case study companies take precedence over the potential benefits related to leveraging the distances in GSD. The benefits of GSD are more to the benefit of the corporations than to the teams or individuals. This leads to the finding that corporate benefits are being pursued in face of the needs of software development teams. While the corporation as a whole is trying to maximise profits for its owners/shareholders, the findings show that the globalised context is generally not to the benefit of the software teams.

Following on from this, it is proposed that the potential benefits may be classified into whether they are to the benefit of the corporation, the organisations (covering the teams of software developers), or to the process of software development. This classification is described as follows:

1. Benefits for the corporation: In findings broadly similar to previous studies, both Pennysoft and Semicon participate in GSD in order to save on costs. That is the fundamental driving factor for these companies, in the pursuit of maximising profits for their shareholders/owners. As such this benefit manifests itself at corporate level, being driven by the overall needs of the company, or indeed of its owners. Having proximity to the target markets and customers is also a commercial benefit, seeking to maximise returns from companies’ clients. Time zone effectiveness has also been cited as a benefit as it may help reduce the time to market for products, thus reducing costs and improving its chances in the
marketplace. Finally, corporations may pursue innovations arising from GSD in order to improve its offerings to customers.

2. Benefits for the organisational units: Other benefits apply more to the workings of the software teams themselves, and as such relate to organisational units as created by the company. GSD offers the benefit of accessing an abundance of skilled labour required to get the work done, which can be of benefit to every team as they are responsible for completing work using available resources. Teams may also act flexibly, re-assigning higher skilled workers to more important core work. Time zone efficiency is claimed to improve the workings of team members as limits that amount of coordination time with colleagues. Geographical distance can help define team boundaries, with each team becoming more independent in the type of tools and processes they utilise and practice.

3. Benefits for the process of software development: The rest of the potential benefits relate to potentially improving the software development process, or the inner workings of each team at an individual programmer level. The distances are claimed to improve the focus on the teams’ processes and documentation. GSD may promote good task allocation according to components of the systems being developed. These processes themselves may be improved by increased innovation between global software teams.

Table 7-1 assigns each of the potential benefits to this proposed classification:
Table 7-1. Proposed classification of the potential benefits of GSD.

<table>
<thead>
<tr>
<th>Corporation</th>
<th>Organisation</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Proximity to market/customer</td>
<td>• Access to large skilled labour pool</td>
<td>• Improved focus on documentation</td>
</tr>
<tr>
<td>• Access to cost-effective labour pool</td>
<td>• Allocation of roles and team structure</td>
<td>• Record kept of asynchronous communication</td>
</tr>
<tr>
<td>• Time zone effectiveness (faster time to market)</td>
<td>• Time zone efficiency</td>
<td>• Cross-site modularisation of work</td>
</tr>
<tr>
<td>• Innovation</td>
<td>• Increased autonomy</td>
<td>• Improved process definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Innovation and shared best practices</td>
</tr>
</tbody>
</table>

Regardless of the realistic ability of companies to achieve all of these benefits in Table 7-1 above, this study has shown that there is a divide between the corporate benefits and the organisational and process benefits. Although the findings show that some of the benefits classified as organisational and process are being realised by each of the companies, they are only by-products of the more substantive benefits at corporate level, cost savings being the most important of all. It is important for us to note that while many of the benefits have been simply mentioned as ‘nice’ aspects of GSD by previous studies, the findings show that: 1) such benefits are not motivating factors for globalising software development, and 2) many such benefits are simply not being realised. Take for example the benefit of increased innovation given the global nature of GSD teams. While such a benefit does sound ‘nice’, neither case study company found it to be important and neither company is realising that benefit.

Indeed, the global environment for software development has been found to be far from perfect for the teams involved. Despite some of the organisational/process benefits being realised, the distances involved introduce much overhead that would not be evident in purely co-located situations. Oshri et al. (2009) predict that such overheads will indeed persist, as companies will continue to struggle to leverage offshoring in face of the complexities involved. Software development is being forced into a distributed and global context in the pursuit of corporate goals. The software development practitioners are being ultimately challenged to adapt to market-driven forces, which is only natural in a commercial context. Thus, it is inevitable for the software development process to be compromised for the ultimate benefit of shareholders. This will be the case as long as
transferring software development to cheaper locations is, or is perceived to be, the cheapest option available to companies.

7.6 Discussion of the benefits of GSD

For each of the potential benefits of GSD, it is important to put them in context alongside previous research findings. This part of the chapter discusses each of those benefits in reflection of the findings from the multiple-case study. An earlier discussion of the benefits was published as Ó Conchúir et al. (2009).

7.6.1 Proximity to market and customer

Semicon use GSD to locate part of their software development closer to Asian markets, as previously noted by Herbsleb and Moitra (2001). However, Pennysoft instead use GSD to move development to cheaper locations, while keeping some software development activities close to their traditional customers. Falling trade barriers have facilitated large scale GSD activities that may not have been possible in the past (Greenhouse 2003). Meanwhile, it was recently found that establishing a captive centre is more expensive for a multinational than it is for a local vendor (Kotlarsky and Oshri 2008), showing the significant investment needed to establish a proximity to market and customer.

7.6.2 Improved focus on documentation

The findings don’t support the proposed benefit as described in Chapter 3. It was not found that a situation that emphasises a focus on documentation is beneficial for practitioners. Instead, teams were forced to put more effort and rigour into formal documentation and processes to improve team communication, a finding that reflects that of DeLone et al. (2005). In relation to this finding, Rönkkö (2007) discusses the mere inadequacy of words and text found in requirements documents used by a distributed organisation if the context and history of that document is not known by the receiver. He points to the increased possibility of challenges to understanding requirements produced in such an organisation context due to the lack of context held by all participants who are coming from different background. Personal communication may be required with the
author to understand what the author really intended, exposing those involved to the coordination complexity in distributed contexts.

7.6.3 Record kept of communications

This lesser-referenced potential benefit was mainly rejected by the findings. The benefit proposed by Damian and Zowghi (2002) is greatly outweighed by the preference of developers to communicate directly and synchronously, and preferably face-to-face.

7.6.4 Access to large skilled labour pool

Semicon claim to have access to the “cream of the crop” with respect to skilled resources. This lies in with Baker et al.’s (2004) claim that GSD allows the organisation to tap into the “biggest brains on earth” to collaborate on complex projects. Pennysoft also has access to skilled resources in India. However, there is a mismatch in Ireland’s expectations of the skills of the Indian developers. The findings run counter to Moitra’s claim (2001) that the process maturity found in India is acceptable. Unlike Pennysoft, perhaps outsourcing-centric companies such as Infosys do possess strong process maturity.

Attrition is a risk that threatens the success of the realisation of this benefit for both companies. Attrition remains a huge concern for Pennysoft who employ in both Bangalore and New Delhi, a phenomenon that was documented by Kobitzsch et al. (2001). Similar concern was expressed by Semicon interviewees. When employees leave a project, the project suffers knowledge loss and a potential interruption to service delivery (Deloitte 2005). High employee turnover can result in costly recruitment campaigns and more effort to train replacement employees. Coupled with this is the difficulty in finding resources with specific technology skills, an issue described at Pennysoft. This issue has been highlighted by Vijayan (2003) who stated that it can be difficult to find skills related to second-tier software packages and technologies, even if there exists a large pool of computer science graduates. An interviewee in Pennysoft feared that the rate of attrition would not reduce in the future, meaning that there may always be a problem with achieving sufficient business expertise in India. However, the Semicon case is at odds with this opinion, as their Indian and Malaysian sites have indeed achieved business maturity.
Neither company agreed that they are pursuing GSD due to lack of availability of skills in their home employment markets, as proposed by Herbsleb et al. (2000) and Nicholson and Sahay (2001). However, it could be argued that pressure on the home employment markets has contributed to increases in wage demands, thus making offshoring more attractive. On this note, it could be argued that a shortage of workers was not generally experienced, but rather a shortage of highly-affordable skilled workers. As for the organisational flexibility afforded by large numbers of skilled employees (DiamondCluster International Inc. 2005), Pennysoft has been experiencing a ramp-up period of at least two years for new offshore teams.

7.6.5 Access to cost-effective labour pool

The Pennysoft case, in particular, brought to light that the company has difficulty in measuring the success of its GSD operations because it’s difficult to estimate the effectiveness of GSD teams. This is because it is difficult to model the cost of GSD. As Nicholson and Sahay found with respect to offshore outsourcing specifically, there may be significant ‘intangible’ costs (Nicholson and Sahay 2004). Interviewees questioned the ultimate cost savings being realised by the company due to the extra effort required to run a GSD project. While accountants can easily argue the case for leveraging cheaper resources, it is more difficult to estimate whether or not the resulting software systems are of the same quality, but also that the use of cheaper resources calls for intense effort from the experienced resources in the higher-cost locations. With respect to software quality in distributed development projects, it was found that during the development of the Microsoft’s Windows Vista operating system that geographical distance had a lesser negative effect on software quality than organisational distance had (Bird et al. 2009). Bird et al. argue that it is more advantageous to promote ‘organisational compactness’ rather than reducing geographical distance in order to maintain acceptable levels of quality.

Offshoring decisions need to be backed by careful analysis that considers full costs and benefits (Roztocki and Fjermestad 2005). Traditional value chain analysis cost estimates (based on overheads and direct labour hours) used by many organisations fail to estimate the true cost of in-house and offshore operations (Roztocki and Fjermestad 2005). The estimations are less likely to identify the ripple effect on the other activities in the organisation’s value chain and may fail to identify hidden costs. Total cost of ownership (TCO) is “a holistic view of costs related to IT acquisition and usage at an enterprise level” (Cappucio
et al. 1996) – a wider view than simply the monetary exchange as agreed upon in the outsourcing contract, or the recruitment costs when setting up a new offshore subsidiary.

The cost savings related to GSD have also been called into question by others. Carmel (2006) argues that companies attempting to implement GSD strategies soon realise that time zones introduce additional coordination costs that negate some of the labour savings. As previously identified, Cockburn (2002, p. 81) found that “project costs increase in proportion to the time it takes for people to understand each other”. In a related finding, Dibbern et al. (2008) showed that projects requiring the client to pass on specialised knowledge to the development organisation resulted in higher costs in offshore outsourcing. The study also showed higher costs arising from geographical and cultural distance, along with high rates of attrition. As previously cited, a report by Deloitte (2005) stated that “real-world experiences suggest that the potential cost savings has been overstated”. Significantly, little research data have been published to indicate actual cost savings due to GSD (while companies may have their own data on GSD projects but may be reluctant to make that data available publicly). CIO Magazine surveyed 101 companies conducting offshore outsourcing, with an average spend of US$16.2 million on outsourcing contracts (Overby 2003a). The survey reported a best-case scenario of 15.2 percent of total added “hidden” costs, on top of the stated contracted price. The worst-case scenario would increase the cost by 57 percent.

The predictions found in the literature resonate with the findings of this multiple-case study. The basic premise of the companies that cheaper salaries offer overall cost savings may not be entirely founded upon actual cost savings figures. The potential fault with this premise is that employing cheaper labour creates global distances between resources who are attempting to build the companies’ software systems. Cost savings depend on the maturity of the organisation and the type of activities involved (Global Insight (USA) Inc. 2005). Hayduk (2003) concludes that cost savings of twenty five percent can be achieved only after one to two years for long-term projects with at least forty team members.

7.6.6 Cross-site modularisation of work

Elements of this benefit have already been discussed in relation to task allocation. While Semicon actively pursue this benefit, Pennysoft argued that their teams are generally too small to realise effective modularisation of tasks.
Kotlarsky et al. (2007) specifically addressed the issue of component-based development in distributed setting. Whilst this thesis has shown two different instances of approaches to cross-site modularisation, Kotlarsky et al. propose perhaps a mid-way approach. The authors directly examine the effects of component-based development on knowledge sharing between distributed sites. They propose that component-based development be adopted (as described in the Semicon case), although it is proposed that components are built across sites rather than used to divide work between sites.

Whilst Semicon adopted their approach in order to minimise daily cross-site communication, Kotlarsky et al. find that such as approach “may also lessen opportunities to share knowledge between sites and may hamper opportunities to reuse existing components”. This approach may overcome a limit to Semicon’s development, that is their difficulty in re-using code and components since each site is quite isolated. Their approach also leads to ‘big-bang’ integration efforts. Kotlarsky et al. argue that knowledge integration and knowledge sharing are imperative for efficient code re-use, and so they argue that it is best to develop components in a cross-site manner. While the authors recognise the costs of cross-site coordination, they don’t cover the topic in detail. This may indeed be a flaw in their recommendations, in light of the findings of both cases in this study that such coordination is highly costly. In a related publication (Kotlarsky et al. 2008a) the authors recommend frequent cross-site communication and centralised tools. This area of component-based development in GSD may need to be studied in further detail to better understand the nuances of how to best approach component-based development, and how different approaches would affect the realisation of the potential benefits of global software development.

7.6.7 Improved process definition

The findings show that both companies generally agreed with Battin et al.’s (2001) proposition that common processes must be applied across sites when cross-site coordination is involved, although there are examples in Pennysoft of the lack of such processes. Strong processes are certainly important within the distributed environment (e.g. Heeks et al. 2001). Pennysoft had trouble in propagating processes across all sites, perhaps due to the “not invented here” syndrome, as referred to by Boutellier et al. (1998). Interestingly, Prikladnicki et al. (2003b) concluded that a strong process definition is an enabler for realising advantages of distributed teams. Overall the findings are at odds with
those findings of Gumm (2007) where improved process definition was identified as a useful benefit arising from GSD. There was a lack of evidence of improved process definition being a benefit of GSD.

7.6.8 Time zone effectiveness

The model of follow-the-sun development promoted by Carmel (1999) was largely rejected by the findings. Ramesh et al. (2006) reported that for the companies in their case studies, the potential benefit of 24x7 development “was far from reality”. Carmel (1999) acknowledged that the follow-the-sun model does not seem to be the main driving factor of companies globalising their software development. Carmel reported on case studies where less than 20% of projects even considered follow-the-sun, while the approach seems only to suit the specialised functions of bug fixing and customer support (Carmel 1999, p. 33).

Espinosa, DeLone and Lee (2006) reported of a company that found that follow-the-sun development was made easier when tasks were loosely-coupled. They note that this can appear to be contradictory, as the follow-the-sun model generally seems to require tight dependency between multiple locations. They found that follow-the-sun was possible when dependencies between tasks could be minimised, or when the management of task dependencies could be simplified and automated as part of the workflow. However, such loosely-coupled work was deemed by Semicon interviewees to render obsolete the potential benefits of temporal distance. Therefore, the findings are at odds with that published study.

7.6.9 Time zone efficiency

The findings agree with Espinosa and Carmel (2004) and Gumm (2007) insofar as the level of direct cross-site coordination activities are reduced when it is not possible to communicate with a remote site. However, the companies in this multiple-case study rejected this as a potential benefit, as direct synchronous communication is required to coordinate effectively. Again, Semicon reduce the need for individual teams to coordinate with other sites, but this takes much forward planning, and added effort at the integration phase.
7.6.10 Innovation and shared best practices

This potential benefit specifically relates to the cultural distance between team members, given the “richness” the differences can offer (Dubé and Paré 2001). While several interviewees in the multiple-case study agreed that cultural distance may lead to a richer variety of approaches to problems, the findings show that innovation is not freely flowing between different sites. This finding relates to Rottman’s (2008) argument that structural dimensions are “boundaries across which social capital must spread for knowledge transfer to take place”. Paradoxically, it may be the organisational distances between team members that are hindering the flow of knowledge of best practices that result from socio-cultural distance. Rottman’s suggested solution is to set up “well-connected teams” so as to allow such social capital to be shared between sites. Again, neither case study company deemed this to be an important benefit, in contrast to claims in previous publications such as by Carmel and Tija (2005).

The findings for Pennysoft showed that there is a lack of respect between sites, while the co-located teams in Semicon limit the amount of cross-site interaction that may lead to sharing of innovation and best practices. Levina and Vaast (2008) found in their study that when the acknowledged status of offshore developers was increased, then they “felt more willing and able to share their ideas while onshore partners were willing to listen”. In light of this multiple-case study, that finding shows that the inter-site collaboration in Pennysoft may be improved if the Irish site were prepared to openly acknowledge the strengths of the Indian site.

Some threats to this benefit have also been identified elsewhere. For example, Casey and Richardson (2004) found that higher-cost workers may not be inclined to share knowledge with the lower-cost counterparts due to the feeling of being threatened, resulting in an ‘us versus them’ attitude, a threat also identified by Herbsleb and Grinter (1999a) and Levina and Vaast (2008). Management at both companies in the multiple-case study were conscious of the possibility of this sense of threat, although the perceived threat may not be felt as strongly as earlier during that decade. Furthermore, the skills and talents of offshore colleagues in lower-waged locations may be underestimated (Grinter et al. 1999, Battin et al. 2001). With a lack of respect for others’ abilities, it is less likely for them to learn best practices from others, and this is what was found at Pennysoft.
While this proposed benefit is promising, it remains to be seen how innovation can be allowed flourish in a GSD context.

7.6.11 Allocation of roles and team structure

The large numbers of graduates found in newly developing employment markets offers flexibility in ramping up development efforts based on organisational and market demands (Battin et al. 2001, Herbsleb and Moitra 2001). Pennysoft do benefit from being able to re-allocate resources as they become available, regardless of their geographical locations. However, as previously described, the case goes against claims of being able to quickly ramp-up offshore teams in days or weeks (Carmel and Tija 2005, p. 11). It was also found at Pennysoft that it is not realistic to simply re-allocate higher skilled programmers to high-value work (DiamondCluster International Inc. 2005), because experienced resources must be assigned for prolonged periods of time to train in new offshore colleagues in their place. It was noted in Pennysoft that it takes time to train in the re-allocated resources, an overhead that is not taken into account by higher-level management who see people as interchangeable “widgets”. This abstraction of people to widgets, to simple producers of work, ignores the emotional and inspirational side of people (Nicholson and Sahay 2001). In the end, this may negatively affect the efficient workings of the organisation if it leads to a high rate of staff turnover.

7.6.12 Increased autonomy

When Gumm (2007) identified this potential benefit, it was also stated that increased autonomy brings with it increased focus on synchronisation of tasks. This is indeed the case at Semicon, where teams can work relatively independently but must concentrate on synchronising their efforts before and after the main development work. Again, Pennysoft do not realise this benefit, particularly for the newer Indian teams. A study by Levina and Vaast (2008) uncovered similar issues to those found in Pennysoft in relation to this potential benefit. Specifically, they found that while the offshore developers were highly qualified, those teams lacked the experience of delivering commercial-grade development projects. Kotlarsky and Oshri (2008) note that companies can make a compromise on reduced costs by targeting countries with matured business expertise, such as that found in Israel. In Pennysoft’s case, it is plausible that autonomy would increase if and when that
site matures with relevant business expertise, similar to the improvements of autonomy noted in the study by Levina and Vaast.

7.7 Conclusion of chapter

Following on from the cross-case analysis of the previous chapter, this chapter has discussed the emerging findings of the multiple-case study at Pennysoft and Semicon. This chapter has described how GSD is being localised in different ways in the two cases. The approach in Pennysoft, similar to virtual teams, contracts the temporal distance between sites by applying pressure on employees to shift their working hours. Temporal distance has been far from eliminated given the interactions between three continents, and the teams may still try to realise the related benefits. However, rather than leveraging this distance, teams generally attempt to reduce it in order to allow for synchronous cross-site communication. Pennysoft localises their software development by designing each team to be made up of co-located members allowing for direct face-to-face communication.

Further to this, the chapter has proposed that team distribution, team size and team maturity are contributing factors to what extent each benefit can be realised in practice. Task allocation is an important factor, as it defines when people need to coordinate their efforts across global distance. A classification of the benefits is proposed based on whether they are intended to positively affect the corporation, organisational unit, or software development processes. In effect, software teams are being forced to work in distributed contexts to satisfy corporate policies. It is ultimately up to the software development practitioners to deal with the globalised reality.

With the analysis of the tension between corporate and other benefits, this study suggests that no combination of beneficial factors of GSD will ultimately outweigh the preferred setting for practitioners of developing software at one location. However, it is the corporate decision makers who should weigh up the potential corporate benefits against the overheads experienced by distributed software projects when deciding on the suitability and configuration of GSD.
Chapter 8. Conclusions

8.1 Introduction
The purpose of this chapter is to highlight key aspects and findings of the research study. The chapter begins with a review of the research questions that drove the study, and then the approach that was taken to address those research questions. The limitations of the study are identified. The contributions and implications of this study for both researchers and practitioners are discussed, and finally potential areas of future research are suggested.

8.2 Review of the research objective
This study was initiated on the basis of a relative paucity of research into the phenomenon of GSD. Researchers have been examining this area for a decade and more, yet there has been a relative lack of attention paid towards the motivations underlying the push to globalise software development. This research study has sought to address this void through a qualitative study of practitioners in two large multi-national companies. Specifically, the following research questions were posed:

- **RQ1**: Identification and framework classification of potential benefits of GSD
- **RQ2**: Investigation of the extent to which these potential benefits are being realised in practice, and the underlying reasons for realisation/non-realisation

The first research question was investigated because of the need to identify a range of potential benefits of GSD as found in previous studies. The benefits, along with the challenges of GSD, were then classified within a framework. Chapter 2 presented this classification, while Chapter 3 dealt with these potential benefits in more detail to better characterise them in terms of the findings of previous studies. The second research question was addressed through the multiple-case study in which practitioners were invited to discuss their experiences with realising each of the potential benefits. They were also asked to reflect on why identified benefits are not being realised within their company.
8.3 Review of research approach

To address the research questions, the multiple-case study was performed involving Pennysoft and Semicon. The study was an exploratory, interpretive, qualitative one. As previously laid out, GSD in itself is a well established research area, although an exploratory approach was deemed relevant for studying the realisation of the potential benefits of GSD. The researcher found that relatively few studies addressed this particular subject. The first step to this exploratory study required setting out the framework of Chapter 2 in order to enumerate the potential benefits as had been already identified by other researchers. The exploratory approach was adopted because it promotes understanding in new research fields where relatively little is known about the nature of the phenomenon (Patton 1990).

Activities associated with the case study included informal heads-up meetings at the companies, the semi-structured research interviews, university-hosted GSD workshops, and research findings feedback sessions. Activities involving the case study companies spanned over three years, from March 2005 to April 2008. Data collection was based on a total of 25 interviews with practitioners. As this was an exploratory study, the interviewees were selected from a variety of roles within the organisations, ranging from a software program manager, to project managers and software engineers. The interviewees were also selected from a variety of projects within each organisation, all of which involved distributed software development. While the majority of the interviewees were based in Ireland, the interviewee sample also included three Indian-based interviewees and one US-based interviewee.

The framework presented in Chapter 2 acted as the basis for the research activities that followed. An interview protocol was devised based on the framework, and it also acted as a lens through which the research data were analysed. The analysis of the multiple-case study (Chapters 5 and 6) explored how the companies deal with realising (or not realising) each potential benefit. Emerging themes arose from the process of analysis, and the discussion of these themes helped to add a richer understanding of the phenomenon.
8.4 Limitations of the research

This study sought to better understand global software development, which is indeed a global phenomenon. The interviewees who participated in the multiple-case study were fully involved in this global context. Nevertheless, the focus of the study was on the Irish sites of the multi-national companies, with all but three of the interviewees being based in Ireland. As the study has focused on this slice of participants in the global phenomenon, the researcher acknowledges that the findings are biased towards the views of these participants, and must be interpreted as such.

A related limitation is that the researcher did not have access to top US-based management who were responsible in both cases for the corporate policy-making of globalising the software development activities. The interviewees of the multiple-case study were describing a situation which was put in place, often due to the policies of others. Their employment depended on US-based corporate policy to allow for software development to take place globally. This means that the responses given during interviews were based on the interviewees’ perceptions of a phenomenon in which they partook, but did not ultimately control. Their opinions and experience were based on their perception of decisions made by others.

Through thick case description, this exploratory study has sought to offer deep insights into a specific research area within GSD. Two unique cases were at the focus of the study. The analysis of the interview data has provided a deeper understanding of the benefits of GSD. The findings can now be applied to similar situations to better understand those contexts. Chapter 4 has addressed the issues of generalisability, reliability and validity.

While manifestations of socio-cultural distance were discussed by many of the interviewees, this distance did not greatly feature in their perceptions of why benefits were being realised or not. Given that socio-cultural distance can affect the understanding between two people communicating with each other (even if they speak the same language natively), the researcher believes that further studies may uncover a deeper insight into how it affects the realisation of the benefits of GSD. If well-managed, this distance may indeed provide many more potential benefits to software development practitioners.

In this multiple-case study, the uniqueness of corporate, team and project characteristics of each case render valid comparison and theoretical generalisation more difficult. Specifically,
interviewees ranged in their roles, length of experience, the nature of distribution of their projects, and the nature of the systems they were developing. To this end, the researcher invited the interviewees to draw from their experiences in past roles and experiences, rather than focusing uniquely on their current project. This approach suited the exploratory nature of the study, as it allowed for interviewees to express their impressions of the phenomenon taking into account all of their previous experiences with it, rather than being constrained to discuss a single project.

With respect to data gathering, a limitation with the interview technique is that the interviewer may introduce bias by their approach of guiding the interview, of word emphasis, tone of voice, body language, and question rephrasing (Cooper and Schindler 2003). While the interviewer must be aware of their effect on the data being gathered, this limitation is accepted as inevitable by interpretivists.

The interview protocol and subsequent analysis of interview data were driven by the GSD framework presented in Chapter 2. The framework takes communication, coordination and control as the underlying processes of software development, and identifies the distances of geographical distance, temporal distance and socio-cultural distance. Other theories may indeed propose different processes and factors to be more important. Furthermore, the GSD literature is not convergent in defining the distances involved, and so the framework and subsequent data gathered may have been significantly changed by defining the distances at a different level of granularity. For example, socio-cultural distance may have been explicitly separated into national cultural traditions, natural language, and organisational culture. On the selection of the twelve potential benefits as identified by the framework, the researcher attempted to include all potential benefits as referred to by existing literature while designing the empirical study. The researcher acknowledges that the importance and breadth of the different potential benefits identified range vastly between corporate-level benefits, down to potential benefits for the individual programmer. While this aspect of the framework allowed the researcher to explicitly conclude that different benefits do indeed apply to different contexts, there is perhaps a fine line between what set of potential benefits should have been considered for the study. As pointed to in a following section, the enumeration of the benefits does provide future studies with a platform from which to further study the set of potential benefits of GSD. Furthermore, given the various weightings of the potential benefits, the researcher
acknowledges that several benefits were included having received minor citations, while other benefits had been cited in many major studies.

8.5 Contributions and implications

This section summarises the new knowledge that has been contributed by this study to the field of GSD, and its implications for both researchers and practitioners.

8.5.1 Contributions and implications for research

The contributions of this research study to the field of research are each outlined as follows.

1. **Enumeration and characterisation of potential benefits.** This study contributes to the research field by providing a list of the potential benefits of GSD and by characterising each of those benefits. An earlier version of this contribution was presented as Ó Conchúir et al. (2006). The study has characterised each of the benefits in terms of identifying what is generally meant by each, to what extent they have been reported to have been realised, and questioning the motivations behind each. The thick description of the findings from both cases and the resultant cross-case analysis provide an in-depth characterisation of each benefit on its own merits. The proposed list of benefits may act as the starting point for further research into the realisation of the benefits.

2. **Framework of GSD.** The structuring of the framework of opportunities and challenges of GSD contributes to the research field of GSD by proposing a relatively simple yet overarching approach to characterising the phenomenon of GSD. The researcher co-presented an earlier workshop version of this framework as Ågerfalk et al. (2005). By building on the fundamental characteristics of GSD that have been proposed by others, the framework succeeds in breaking down the core components of GSD to provide an over-arching view of it. While some items placed within the framework do cover multiple cells, the format works relatively successfully in identifying the key opportunities and challenges of GSD. Furthermore, each cell of the framework may be used to further investigate the area. For example, the cell in relation to coordination and temporal distance may be
further investigated (such as the work by Espinosa and Carmel (2004)). Indeed, while the framework may be used in future studies, it may also be further refined. For example, while the framework uses three distances to categorise existing knowledge, on-going studies are helping to improve our understanding of these dimensions. Gumm (2006) has attempted to describe the very nature of ‘distribution’ in GSD, while O’Leary and Cummings (2007) granularised distribution by defining different permutations of geographical distance within teams.

3. **Insight into current state of practice.** The findings provide an insight for researchers into how companies are currently realising the benefits of GSD. The study describes to what extent the benefits are being realised, and practitioners’ views on them. The findings offer motivation to further understand how best to realise the set of benefits, as they confirm that these large organisations are finding it difficult to leverage all aspects of GSD. The study has described for both companies their motivation for practicing GSD, how they structure their teams, and how they approach the coordination problem of developer software in a distributed manner. The insight into current working practices of the two different companies has shown that their working practices affect their realisation of the benefits, and this implication is further discussed below.

4. **Localisation of GSD.** The study has described the tensions between the benefits and challenges of GSD in relation to geographical, temporal and socio-cultural distance. For researchers, the question remains of how software development should be structured so as to best leverage the benefits of the distances while acknowledging the difficulties of GSD. This study has described the tension between the globalising of software development and efforts to minimise particular distances within projects. Dubé and Paré (2001) characterised distributed software development teams according to how “localized” they were. The findings of this study have shown that work practices in both case study companies are designed to ‘localise’ certain aspects of the GSD projects. Specifically, Pennysoft work in a way that reduces temporal distance between different sites by shifting their working hours. Geographical distance and cultural distance remain, as the projects are still distributed globally. Semicon have limited the distribution within their projects, and also each team is made up of individuals at one site only. In line with a best practice
recommendation by Carmel and Agarwal (2001), this type of localisation reduces the effects of temporal distance on daily work tasks. It also implicitly decreases the level of socio-cultural distance given that a majority of a team’s members will typically come from that country. In relation to the benefits of GSD, this finding shows that potential benefits resulting from the global distances (such as improved documentation and process, and improved innovation) are being disregarded in the face of the difficulties of developing software in a distributed context. The findings on localisation emphasise the need for synchronous communication between two co-dependent individuals. Future studies will have to contend with the tension between pursuing the full range of benefits of GSD while contending with coordinating across distance.

5. **Overheads and the benefits of GSD.** It has been widely reported that much of the globalisation of software development is as a result of pressures to reduce costs. The two case study companies are no different. Participants identified this as the primary motivating factor behind the GSD activities. Although this study has not provided a cost-benefit analysis grounded in economics, the findings have shown that the very approaches adopted by the companies introduce coordination overheads. These overheads are the product of the need to coordinate across distances. Even in Semicon’s case where teams have been localised to reduce coordination overheads, there are separate overheads in planning and integrating the development of system components. This study has pulled closer together claims of benefits of GSD with the challenges of working in a distributed manner. Future studies should build on this to study how coordination overheads can best be managed while maximising the realisation of each of the potential benefits.

6. **Effect of team characteristics and task allocation on realisation of benefits.** This study has offered a new perspective on the realisation of GSD benefits, that is that different approaches to structuring GSD teams will directly affect what benefits can be realised. There has been considerable research conducted into the distribution of teams and the allocation of tasks for such teams. However, this study has combined that research with the previously disjointed phenomenon of the benefits of GSD. This finding comes back to how coordination is managed across distance. First, if there was zero dependency between two developers at two different sites, there would never be reason to open any communication channels.
However, a software development project is a complex undertaking and requires adequate coordination of the people working on it. A global software development project will always require some form of global coordination, and how this is managed will affect what benefits are ultimately realised. Notably, Pennysoft’s team distribution makes it possible for them to attempt to leverage benefits in relation to temporal distance, and it enables them to better leveraged dispersed skill sets. Semicon’s team size and maturity, however, allows them to ‘localise’ their development work. This approach reduces the daily effects of temporal distance on their work while leveraging cheaper workforces. This eliminates the chances of realising benefits in relation to temporal distance such as time zone effectiveness, and benefits in relation to socio-cultural distance, such as the spread of innovation and shared best practices. This contribution of knowledge should allow future research studies to further investigate approaches to maximise the realisation of the potential benefits of GSD.

As a conclusion to the contributions to research, the findings of the effects of localisation, team characteristics, and of task allocation help to bring researchers closer to understanding the ‘sweet spot’ of maximising the realisation of benefits of GSD while minimising the negative effects of global distance by localising team structures. This study has concluded that the most important benefits of GSD are strongly aligned with corporate goals. While many elements of GSD may seem to be beneficial, the findings have shown that they are overshadowed by the overheads involved with having to develop software in a distributed context. Given the overwhelming difficulties involved with working across global distances, this study into the nature of the benefits has essentially concluded that these benefits in themselves do not justify globalising software develop for the sake of it. However, it is a commercial reality that companies will continue to pursue the availability of cheaper skilled labour wherever it may be found. As such, researchers should continue to investigate how best to realise the potential benefits within practical constraints.

8.5.2 Contributions and implications for practice

While many of the contributions outlined above may also apply to practice, here are implications specifically for practitioners arising from the multiple-case study.
1. **Tension between corporate and other benefits.** As already pointed to in this chapter, the data of this multiple-case study are perspectives from practitioners who were mainly based in Ireland. The corporate policies perusing the realisation of certain benefits of GSD were quite obviously making life harder for the software development teams. This study has presented the set of benefits according to a classification of ones that apply to the corporation, organisational unit, or software development processes. There is a tension between the corporate benefits that dictate corporate policies and the benefits that software development teams may realise in a distributed context. While it is beneficial for the corporation to be closer to target markets, to access cheaper skills in abundance and to reduce time to market, this does not necessarily translate to software development teams benefiting from the imposed global distances. This study brings to the attention of practitioners that the driving factors behind a GSD effort should first be prioritised, but also the effects on the software development teams should be taken into account. Practitioners need to closely examine the costs of conducting software development in a globally distributed context. Cost-benefit analyses of upcoming GSD projects should take into account the variety of potential benefits on offer, while comparing those against the extra effort that will be required to complete such a project.

2. **Team structure and task allocation.** The findings of this study emphasise for practitioners the role of team structure and task allocation in the realisation of the benefits of GSD. First, it was highlighted in the Pennysoft case that upper-level management viewed individuals as ‘widgets’ that may be simply re-allocated to new projects with no associated ramp-up costs. The findings bring into question previous claims that expanded virtual organisations may be brought online within a matter of days and weeks rather than months and years. Pennysoft are ramping-up gradually in India over several years. Furthermore, it has been shown that the management of cross-site dependencies plays a vital role in the type of development team that emerges, and thus can limit the extent to which particular benefits may be realised. Truly virtual teams can offer a dynamic organisational structure that can adapt globally to include skilled individuals no matter where they may be located. A more localised approach such as in Semicon requires co-located teams with full skill sets and upfront planning, but it may allow for more efficient
day-to-day software development. This complete mix of team characteristics and task allocation must be taken into account by practitioners when assessing which benefits of GSD can be realised.

3. **Realisation of benefits should not be assumed.** Following on from the finding that the realisation of the benefits depends heavily on team structure and task allocation, its implication for practitioners is that the potential benefits are not universally available to any one GSD project (at least given our current understanding of the phenomenon). This conclusion is compatible with Kotlarksy and Oshri’s (2008) finding that companies will most likely fail to benefit from GSD if they simply rely on the economic attractiveness of any particular employment market. Not all team structures will allow for all benefits to be realised. Indeed, neither approach described in the findings allow for all of the potential benefits to be realised. It can no longer be assumed that if a team manages to overcome all challenges related with GSD that they will achieve the full range of GSD benefits. For this reason, the results of this study should motivate practitioners to better analyse which benefits at corporate, organisational and process levels they wish to realise, and then to decide how best to realise them.

4. **Cost versus quality.** It was found that policies to globalise software development efforts can be enforced by upper-level managers who are not directly visible to the participants of the study. For example, in Pennysoft approximately 40 percent of work was to be sent to lower-cost locations in Asia, regardless of the practical difficulties of applying this policy. Ironically, it was found that these policies that are targeted at cost cutting are in themselves adding overhead costs. In Semicon, the costs manifest themselves as the face-to-face contact required for planning and integration phases to allow distributed project teams to work on system components. In Pennysoft, it was found that the ramp-up of the Indian site is taking in excess of two years. During that time, Irish managers must coordinate daily with the Indian teams, and Irish developers must spend time in assisting less experienced developers in India. These findings contribute to previous studies that have identified ‘hidden’ costs in GSD. The coordination costs in GSD can be significant, and will affect the upfront cost savings of cheaper salaries and infrastructure. Again, approaches to manage such coordination costs will themselves determine which potential benefits may be realised.
As a conclusion to the contributions to practice, this study has provided practitioners with a detailed characterisation of a certain aspect of GSD – that is, its potential benefits. The lack of such a nuanced organisational understanding of the benefits was evident at the case study companies. With a structured list of these potential benefits, organisations can better match their strategic goals with how they structure their GSD activities ‘on the ground’. Given this data, organisations can no longer simply assume that successful GSD projects result in all of their intended strategic goals being met.

8.6 Potential areas for future study

There is much potential for further research in the field of GSD, as current studies are still attempting to characterise the phenomenon and understand its implications for software development. This section focuses on the potential for future research specifically related to the benefits of GSD.

GSD employs a wide range of practitioners around the world. This thesis described a study that focused on practitioners in two multi-national companies who were based in Ireland. A similar study could offer further unique views on the phenomenon by narrowing the unit of analysis (such as a single project) while studying its practitioners across all of its locations globally. The area of focus may also be narrowed, compared to that of this exploratory study. For example, a similar study may focus on different team configurations and their relation to the realisation of the benefits. Team maturity is another factor that was highlighted by the findings of this study – future longitudinal studies may examine team maturity over time in relation to which benefits are being realised.

The tension between corporate, organisational, and process objectives in the pursuit of realising the benefits of GSD is another interesting area that may warrant further research. The interview process of this study made it very clear that different levels of the organisation can have rather separate objectives in relation to GSD, and the application of those objectives affect various parts of the organisation. The corporation may be striving towards cost reductions, whilst the organisational structure may attempt to leverage temporal distance, and at the same time the individual engineers are being constrained by being separated from their colleagues by several thousand kilometres.
Another field of interest is what this study has described as the ‘localisation’. Is GSD best realised when one or more of the distances are minimised? What distance should be minimised? At what cost can the distances be reduced? What benefits are maximised or reduced depending on different localisation approaches? A future study may also focus on the reduction of socio-cultural distance which did not strongly come to light in the findings of these two cases. In part, researchers have already been addressing this aspect of the phenomenon by attempting to understand how to minimise the effect of the global distances involved. Moreover, there is much opportunity for studies to further address the leveraging of the inherent global distances, rather than their suppression, in order to realise a wider range of benefits.

8.7 Final reflections

Software development is a difficult process. In the researcher’s opinion, its complexity can be often overlooked or not fully comprehended. The nature of software development has never been static. Programming languages are being defined, and existing languages are evolving. Approaches to systems development sway between highly formal and heavily-documented processes, and light build-as-needed agile processes. Communication tools are evolving. Commercial systems development is inherently dominated by the need to survive against competing products with innovative features while controlling costs. With this in mind, global software development in itself is far from being a static phenomenon. Companies will continue to push the limits on how software systems can be developed to maximise return on their work. Market forces requiring either the cheapest labour or the most skilled labour will continue to play out against each other. Along with this, advances in communication technologies will result in continuous refinements in the nature of approaching distributed software development. We as researchers still do not have the answer as to what is the optimal mix of approaches of distributed software development so as to maximise commercial returns. The research community must strive to understand the very fundamentals of GSD, while continuing to offer innovative approaches to developing software in a distributed manner. There is an exciting future in store for the development of information systems as both researchers and practitioners push the boundaries of how people can collaboratively contribute to development projects no matter where they are physically located.
Appendix A  Interview Protocol

**Interview aim:** To develop an understanding of the GSD business environment, and more specifically, the realisation of the potential benefits of GSD within the organisation and at a practical day-to-day level.

**Respondents:** Knowledgeable managers and developers that have varied experience with GSD projects.

1. Introduction

- How many years have you been employed at the company?
- What is your role?
- Describe your role in relation to the rest of the organisation.
- What does your work mainly consist of?
- To what extent does your work deal with distributed software development?
- To what degree is your team distributed?

2. Benefits and motivation

- Why does your company practice GSD?
- What benefits do you think are expected as a result?
- Who within your company has decided to target these benefits?
- To what extent do you think the benefits are being realised?
3. Communication

Geographical distance

- To what extent do you and your team communicate across distance? How often?
- What is your impression of the effectiveness of such communication?
- What challenges do you see arising from communicating across geographical distance?
- What approaches are taken to resolve these challenges?
- What benefits do you see arising from communicating across geographical distance?
- For each of the following benefits related to communication across geographical distance to what extent are you, your team, your project, and your company realising the benefit? If the benefit is not being realised, why? How do you think the benefit may best be realised?
  - [List each related benefit and related questions]

Temporal distance

- Do you and your team communicate globally across temporal distance?
- In your experience, is communicating across temporal distance an opportunity or a challenge?
- How does temporal distance affect your work?
- What is your impression of the effectiveness of such communication? Why?
- What challenges do you see arising from communicating across geographical distance?
- What approaches are taken to resolve these challenges?
• What benefits do you see arising from communicating across temporal distance?

• For each of the following benefits related to communication across temporal distance to what extent are you, your team, your project, and your company realising the benefit? If the benefit is not being realised, why? How to you think the benefit may best be realised?

  o [List each related benefit and related questions]

[The initially lengthy interview protocol continued to list similar questions for communication, coordination and control across geographical, temporal and socio-cultural distances.]

4. Concluding questions

• Is there any other part of your experience with GSD that you see applicable to this theme?

• Upon reflection of this conversation, what is your impression of GSD with respect to the potential benefits?

• What other benefits do you see as a result of GSD?

• What lessons have you learned in dealing with GSD?

• How do you see GSD evolving in your company?
Bibliography


Herbsleb, J. D. and Grinter, R. E. (1999a) 'Architectures, Coordination, and Distance, Conway's Law and Beyond', IEEE Software, 16(5), 63–70.


Parnas, D. L. (1972) 'On the criteria to be used in decomposing systems into modules', *Communications of the ACM, 15*(12), 1053 – 1058.


