

JULIA KOTLARSKY

# Management of Globally Distributed Component-Based Software Development Projects



**Management of Globally Distributed  
Component-Based Software Development Projects**

Julia Kotlarsky



# **Management of Globally Distributed Component-Based Software Development Projects**

Management van wereldwijd gedistribueerde componentgebaseerde  
softwareontwikkelingsprojecten

## **Thesis**

to obtain the degree of Doctor from the  
Erasmus University Rotterdam  
by command of the  
rector magnificus  
Prof.dr. S.W.J. Lamberts  
and according to the decision of the Doctorate Board

The public defence shall be held on  
Thursday 16 June 2005 at 16:00 hrs

by

**Julia Kotlarsky**

born at Kharkov, Ukraine

**Doctoral Committee:**

Promotor: Prof.dr. K. Kumar

Other members: Prof.dr. L.P. Willcocks

Prof.dr. S.J. Magala

Prof.dr. F.M. Go

Copromotor: Dr. J. van Hillegersberg

Erasmus Research Institute of Management (ERIM)

Rotterdam School of Management (RSM) Erasmus University

Internet: <http://www.irim.eur.nl>

ERIM Ph.D. Series Research in Management 59

ISBN 90-5892-088-7

Cover photo:

©2005, Karin Oppeland

©2005, Julia Kotlarsky

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the author.

## ACKNOWLEDGEMENTS

Many people contributed to this thesis in different ways. First, I would like to thank my supervisors Kuldeep Kumar and Jos van Hillegersberg for supporting me through the PhD process and for leaving me the freedom to pursue my ideas while providing useful comments and guidance.

I would like to thank members of the inner committee, Slawek Magala, Leslie Willcocks and Frank Go, for making time to read this thesis. For their help in other non-thesis related areas, I wish to thank Slawek for advising me during my early year as a Ph.D. student (and for carrying my very heavy suitcase in Florida Keys) and Leslie for supporting my career at WBS while I was finishing my studies (and for the ski turns).

One of the main struggles of PhD students doing empirical research is to obtain access to companies. I was lucky to be able to get access and collect data in very interesting companies. This would not have been possible without the help of several individuals from these organizations, to whom I am very grateful. My special thanks go to Corey Hirsch for opening doors for me at LeCroy and for his hospitality while I was visiting LeCroy in NY; to Alain Lesaffre for arranging my visit to SAP Bangalore and offering SAP's guest house while in India; and to Christoph Thommes for his feedback and responsiveness in providing additional data.

I also thank all the interviewees for their time and information they provided.

At different stages of my Ph.D. trajectory I was fortunate to meet more experienced researchers who gave me advice and feedback on my work. I am very grateful to Magdy Serour for helping me to link different aspects of my research; to Jane McKenzie for our long discussions on patterns emerging from the data; to Paul van Fenema for his ideas and comments on my thesis, and for advising me on different issues throughout my Ph.D.

During my Ph.D. journey I have made remarkable friends. Special thanks go to my two paranymphs Viara and Anna for listening and being around when I needed help and advice. I would like to thank Karin for stimulating me to be strong, for long talks about the philosophy of life, and for presenting me with the photo of dancing ladies that is now on the cover of this thesis. To Rouven, Guillermo, Ira, Daina and Tineke, thank you for sharing with me exciting times of being on the top and for helping me during difficult times, and of course for all the fun times we have had together.

When I arrived in The Netherlands, I did not have any relatives in this country. I was lucky to meet Tatiana and Evert van den Meulen who very soon adopted me as their ‘granddaughter’ (and whom I adopted as grandparents): I am very grateful for your care and love.

Being abroad and far away from my family has made me realize how important they are in my life. I am thankful to my younger brother Pavel, for being, from time to time, my ‘big brother’ when I needed support and advice. I am forever indebted to my parents for the motivation to get me to where I am now, and for supporting me in the decisions I have made.

Half-way through my Ph.D. journey I met Ilan, who became my friend, partner and even my co-author, and whose support has been and is invaluable for me. Ilan, thank you for being my advisor, for listening when I was getting lost in the jungle of empirical data and helping me to find a way out; for motivating and encouraging me through the most difficult times of my Ph.D. journey; and for helping me to see what matters the most.

*April 2005, Rotterdam*

**TABLE OF CONTENTS**

- CHAPTER 1 RESEARCH MOTIVATION..... 1**
- 1.1 Introduction: What is Special About Software Development? ..... 1
- 1.2 The Phenomenon: Globally Distributed Component-Based Development 2
  - 1.2.1 Globalization in the software industry..... 2
  - 1.2.2 Adopting Component-Based Development methodology ..... 3
- 1.3 A Gap in the Literature..... 6
- 1.4 Research Objectives and Question ..... 8
- 1.5 Focus of this Research..... 9
- 1.6 Relevance and Contributions of this Research ..... 12
- 1.7 Outline of this Thesis ..... 13
- CHAPTER 2 LITERATURE REVIEW ..... 15**
- 2.1 Introduction ..... 15
- 2.2 Management of Globally Distributed Software Development ..... 16
  - 2.2.1 Problems and breakdowns in GDSD ..... 19
  - 2.2.2 How to organise and manage GDSD ..... 21
  - 2.2.3 Conclusions from IS research on the management of GDSD projects. 32
- 2.3 Research on Collaboration in Globally Distributed Teams..... 34
  - 2.3.1 Social aspects in globally distributed teams ..... 34
  - 2.3.2 Knowledge sharing in globally distributed teams..... 36
  - 2.3.3 Conclusions from OB research on globally distributed teams..... 39
- 2.4 Management of Component-Based Software Development ..... 40
  - 2.4.1 A Historical perspective on Component-Based Development ..... 40
  - 2.4.2 CBD in the software industry: Background..... 43
  - 2.4.3 A Component-Based system..... 44
  - 2.4.4 The CBD approach: Process overview ..... 49
  - 2.4.5 Reuse in CBD: Benefits and challenges ..... 51



2.4.6 How to organise and manage CBD in a globally distributed environment .....	57
2.5 Success in Information System Projects.....	70
2.6 Conclusion: Potential Success Factors .....	73
<b>CHAPTER 3 RESEARCH FRAMEWORK: THE THEORETICAL LENS FOR EMPIRICAL INVESTIGATION.....</b>	<b>75</b>
<b>CHAPTER 4 RESEARCH METHOD AND PROCESS.....</b>	<b>79</b>
4.1 Qualitative Case Study Research.....	79
4.2 Selecting the Companies and Projects to be Studied.....	82
4.3 Research Design and Process .....	84
4.4 Data Collection: Methods and Process.....	86
4.5 Data Analysis Process and Instruments.....	89
4.5.1 Within-case analysis .....	89
4.5.2 Cross-case analysis .....	98
4.6 Quality of the Empirical Research .....	99
4.7 Conclusions: Case Study Template .....	101
<b>CHAPTER 5 CASE STUDY OF LECROY CORPORATION.....</b>	<b>105</b>
5.1 Background .....	105
5.1.1 Background of LeCroy global organization .....	105
5.1.2 Background of the project and product under study.....	106
5.1.3 Background of the software team .....	110
5.2 Data Collected .....	112
5.3 How LeCroy Organises and Manages GD CBD: Analysis and Results	115
5.3.1 LeCroy concept map.....	115
5.3.2 Factors and managerial practices that contribute to success: Causal relationships .....	118
5.3.3 Managerial practices: Description and evidence .....	120
5.3.4 Success in GD CBD: Evidence.....	138
5.4 Conclusions .....	142

<b>CHAPTER 6</b>	<b>CASE STUDY OF SAP.....</b>	<b>145</b>
6.1	Background .....	145
6.1.1	Background of SAP global organization .....	145
6.1.2	Background of the project and product under study .....	146
6.1.3	Background of the software team .....	148
6.2	Data Collected .....	151
6.3	How SAP Organises and Manages GD CBD: Analysis and Results .....	154
6.3.1	SAP concept map.....	154
6.3.2	Factors and managerial practices that contribute to success: Causal relationships .....	156
6.3.3	Managerial practices: Description and evidence .....	158
6.3.4	Success in GD CBD: Evidence.....	173
6.4	Conclusions .....	177
<b>CHAPTER 7</b>	<b>CASE STUDY OF TCS.....</b>	<b>181</b>
7.1	Background .....	181
7.1.1	Background of TCS global organization .....	181
7.1.2	Background of the project and product under study .....	182
7.1.3	Background of the software team .....	189
7.2	Data Collected .....	191
7.3	How TCS Organises and Manages GD CBD: Analysis and Results .....	194
7.3.1	TCS concept map.....	194
7.3.2	Factors and managerial practices that contribute to success: Causal relationships .....	196
7.3.3	Managerial practices: Description and evidence .....	198
7.3.4	Success in GD CBD: Evidence.....	222
7.4	Conclusions .....	225
<b>CHAPTER 8</b>	<b>CASE STUDY OF BAAN .....</b>	<b>227</b>
8.1	Background .....	227
8.1.1	Background of Baan global organization .....	227

8.1.2	Background of the project and product under study .....	231
8.1.3	Background of the software team .....	233
8.2	Data Collected .....	235
8.3	How Baan Organises and Manages GSD: Problems Faced and Implications for Success Factors .....	238
8.4	Possible Impact of the Adoption of CBD on the Success of the E- Enterprise Project: Discussion.....	268
8.5	Conclusions .....	271
<b>CHAPTER 9</b>	<b>CROSS-CASE ANALYSIS AND RESULTS.....</b>	<b>275</b>
9.1	Similarities and Differences Between the Studied Cases.....	275
9.2	Managerial Practices Perceived as Important for Success: Cross-case Results .....	280
9.3	Factors Contributing to Success: Cross-case Results .....	318
9.4	Conclusions .....	321
<b>CHAPTER 10</b>	<b>CONCLUSIONS.....</b>	<b>323</b>
10.1	Theoretical Lens: Revisited.....	323
10.2	The Role of Technology: Technology Alone is Not Enough.....	325
10.3	How Companies Organise and Manage CBD in a Globally Distributed Environment: Successful Managerial Practices.....	327
10.4	The Role of Context in Selecting Managerial Practices: The Context Does Matter .....	329
10.5	Theoretical Contributions.....	330
10.6	Practical Contributions: Implications and Lessons for Managers .....	332
10.7	Limitations.....	339
10.8	Suggestions for Future Research .....	340
<b>APPENDICES</b>	<b>.....</b>	<b>341</b>
<b>EXECUTIVE SUMMARY</b>	<b>.....</b>	<b>353</b>
<b>REFERENCES</b>	<b>.....</b>	<b>355</b>
<b>CURRICULUM VITAE</b>	<b>.....</b>	<b>367</b>

## LIST OF FIGURES

Figure 1: Research focus .....	9
Figure 2: Related research streams.....	10
Figure 3: Thesis structure and research design.....	14
Figure 4: The Waterfall approach: traditional software development lifecycle ....	21
Figure 5: Scenario A - GDSD organised <i>by phase / process step</i> .....	24
Figure 6: Scenario B - GDSD organised <i>by product structure (product module)</i> .	24
Figure 7: Scenario C – only well-defined tasks distributed across locations .....	25
Figure 8: Scenario C – GDSD based on <i>product customisation</i> .....	26
Figure 9: Potential factors that contribute to success in GDSD projects.....	32
Figure 10: Potential factors that contribute to success in GD teams .....	39
Figure 11: The main concepts behind components .....	45
Figure 12: V-cycle approach: CBD lifecycle .....	50
Figure 13: Division of onsite/offshore work for development of the Skandia platform.....	60
Figure 14: Potential factors that may contribute to success in GD CBD .....	73
Figure 15: Theoretical lens.....	76
Figure 16: Research process and research design .....	85
Figure 17: Phases of within-case analysis .....	90
Figure 18: The data sorting and linking approach.....	93
Figure 19: Case study template .....	102
Figure 20: Division of responsibilities between NY and Geneva offices .....	106
Figure 21: Major chronological phases of the Maui project .....	107
Figure 22: Maui product architecture (schematic) .....	109
Figure 23: Organizational structure of LeCroy software team .....	111
Figure 24: How LeCroy organises and manages GD CBD to be successful.....	117
Figure 25: Organizational structure of KM Collaboration group .....	150
Figure 26: How SAP organises and manages GD CBD to be successful .....	155
Figure 27: Quartz component-based architecture.....	183

Figure 28: Technical overview of Quartz implementation .....	184
Figure 29: Quartz: product development and solution implementation methodology .....	185
Figure 30: Skandia project: Apollo CB architecture .....	186
Figure 31: Dresdner project: Investment and e-commerce bank .....	188
Figure 32: Typical organizational structure of the Quartz implementation project: Skandia and Dresdner projects.....	190
Figure 33: How TCS organises and manages GD CBD to be successful .....	195
Figure 34: Skandia project: onsite-offshore delivery model .....	201
Figure 35: Baan stock prices .....	228
Figure 36: Products included in the E-Enterprise suite .....	232
Figure 37: Organizational structure of the E-Enterprise development group .....	234
Figure 38: Roles (people) involved in the management of each of the eight products comprising E-Enterprise.....	241
Figure 39: Compatibility between versions of different products .....	254
Figure 40: Inter-site coordination: Propositions.....	291
Figure 41: Appropriate tools and technologies: Propositions.....	297
Figure 42: Social ties: Propositions.....	304
Figure 43: Knowledge sharing: Propositions .....	311
Figure 44: Components management: Propositions.....	318
Figure 45: Theoretical Framework.....	324
Figure 46: How companies organise and manage GD CBD to be successful.....	328

## LIST OF TABLES

Table 1: Overview of core literature on management of GDSD projects .....	18
Table 2: Types of Collaboration Technology .....	30
Table 3: Overview of core literature on management of CBD.....	56
Table 4: Main steps in data collection process .....	88
Table 5: LeCroy: Interview and data collection details.....	113
Table 6: Contribution of managerial practices to success at LeCroy .....	119
Table 7: SAP: Interview and data collection details.....	152
Table 8: Contribution of managerial practices to success at SAP .....	157
Table 9: TCS: Interview and data collection details.....	192
Table 10: Contribution of managerial practices to success at TCS .....	197
Table 11: The Rise & Fall of Baan Co. ....	228
Table 12: Baan: Interview and data collection details.....	236
Table 13: Capabilities of SD tools at Baan.....	255
Table 14: Collaborative technologies used in Baan .....	256
Table 15: Would adoption of CBD help to avoid the problems: discussion .....	268
Table 16: Similarities and differences between the studied cases.....	277
Table 17: Managerial practices: comparison of results across cases.....	282
Table 18: Capabilities of SD tools: comparison of results across cases.....	292
Table 19: Requirement for ICT infrastructure: comparison of results across cases .....	294
Table 20: Collaborative technologies: comparison of results across cases .....	295
Table 21: Factors contributing to success (per success dimension) .....	320
Table 22: Factors contributing to success.....	326
Table 23: Checklist for managers.....	334
Table 24: A Guide to Tools and Technologies for managers of GD CBD.....	336



## CHAPTER 1 RESEARCH MOTIVATION

### 1.1 INTRODUCTION: WHAT IS SPECIAL ABOUT SOFTWARE DEVELOPMENT?

Historically the demand for software services has outpaced supply. As we enter the era of e-business, companies are increasingly adopting complex software systems to support their internal and external processes. At the same time, we are also witnessing an exponential increase in the use of embedded software systems. For example, mobile phones, personal organizers and cars are beginning to be equipped with sophisticated software which communicates over the web. Consequently, the demand for software and software developers is exploding in all parts of the world.

The imbalance between demand and supply is further exacerbated by the high levels of skill and training required for building software. Software engineering organizations have always had trouble meeting the growing demand for high quality software (Karolak 1999). Although numerous improvements have been introduced to software engineering practices, Brook's (1987) claim that 'building software will always be hard' is now generally accepted. Brooks (1987) listed four unique properties of software that make software development more difficult than other systems engineering disciplines. Software systems are *complex*, *unvisualizable*, and are constantly subject to *change*. Finally, software systems are expected to *conform* to the continuously changing environment in which they operate. These 'inherent' difficulties of software make software engineering a complex discipline, and consequently, large software development projects are regularly delayed and show huge budget overruns (Willcocks et al. 2002; Wallace and Keil 2004). As a result, highly skilled software engineers and experienced software project managers are scarce and expensive in most regions of the world.



## **1.2 THE PHENOMENON: GLOBALLY DISTRIBUTED COMPONENT-BASED DEVELOPMENT**

### **1.2.1 GLOBALIZATION IN THE SOFTWARE INDUSTRY**

In order to build quality software faster and more cheaply, companies in industrialized countries are turning to globally distributed software development projects.

Emerging countries such as India and Israel are known to have large pools of highly trained software engineers at relatively low cost. Moving parts of the development process to these emerging countries can not only decrease development costs, can also provide access to scarce development manpower and resources (Carmel 1999; Karolak 1999; Sarker and Sahay 2004). Another advantage of global distribution could be reducing project lifecycle by using time-zone differences to organise ‘follow-the-sun’ (or ‘round-the-clock’) development (Carmel 1999; Evaristo and van Fenema 1999; Herbsleb and Moitra 2001).

Global distribution of software development has become widespread over the last decade. There are a number of economic and technical drivers that are likely to further accelerate the growth of distributed software development. For economic and financial considerations, many companies are switching to globally distributed development and/or offshore outsourcing of products and services. For instance, in the software and electronics industries offshore outsourcing of development (in the software industry) and manufacturing (in the electronics industry) is very common. Outsourcing of services such as call centres to English-speaking developing countries is becoming increasingly common. Many companies are opening R&D or manufacturing centres in countries where costs are low and yet skills and expertise are available (e.g. India, China). The recent trend towards mergers and acquisitions can also result in globally distributed organizations. Global distribution is also useful whenever a software product needs to be

customized for a local market. Proximity to the customer may be necessary in such cases (Carmel 1999).

On the technological side, ongoing innovations in Information and Communication Technologies (ICT) increase the possibilities to cooperate in a distributed mode.

However, the geographical distance, time-zone and cultural differences associated with global distribution has caused problems for globally distributed software teams, such as the breakdown of traditional coordination and control mechanisms, commonly used in co-located software teams, asymmetry in distribution of information between dispersed sites, misunderstandings and loss of communication richness (Carmel 1999).

Despite the problems and breakdowns that the people involved in Globally Distributed Software Development (GDSD) projects have experienced, more and more companies are becoming involved in GDSD. Gartner Group predicts: 'Globalization is inevitable, IT groups that plan their responses to the challenges raised by this complex issue have a better chance of succeeding in the increasingly competitive environment of software development' (Lyengar 2004).

### **1.2.2 ADOPTING COMPONENT-BASED DEVELOPMENT METHODOLOGY**

Typically, software systems have a long lifetime, of at least several years, during which such systems are upgraded and enhanced with more features, and released as different versions. Changes, improvements, and enhancements leading to new software design releases cause a software system to evolve (Peters and Pedrycz 2000). As a result, a software system needs to be updated and changed many times over the period of time that a system lives (Brooks 1987; Crnkovic and Larsson 2002). Therefore, the software industry has recently started to adopt a more modular or Component-Based (CB) architecture that facilitates development of software products with a long lifetime.

In terms of system structure, CB system architecture is considered to be a key to the success of systems with a long life cycle (Crnkovic and Larsson 2002). As compared to a monolithic software system, a CB system is considered to be flexible, extensible, and reusable (Crnkovic and Larsson 2002). Furthermore, a CB system is easier and more effective to maintain, because it can be maintained in parts (by components), as opposed to a monolithic system that needs to be maintained as a whole (Verbraeck et al. 2002).

Component-Based Development (CBD) has its roots in manufacturing. The trend to develop products that have *component-based* or *modular* architecture is well established in the automotive, electronics, aeronautic and other manufacturing industries. Since the mid-60s when the concept of modular production was introduced, modular (later referred to as component-based) product architectures have become dominant in several manufacturing industries.

In the software industry, CBD is a relatively new trend. It emerged in the mid-90s with the introduction of software component technologies such as Enterprise JavaBeans, Microsoft COM and CORBA, and is increasingly becoming a major trend in software development (Peters and Pedrycz 2000; Kim 2002).

***Component-Based (Software) Development*** involves (i) development of software components and (ii) building software systems through the planned integration of pre-existing (developed in-house or procured from the component market) software components (Bass et al. 2000).

Initially, CBD methodology was presented as a revolutionary approach to software development, promising dramatic improvements in software development efficiency, such as better quality, shorter time-to-market, and lower development costs (Huang et al. 2003; Vitharana 2003). The main advantage expected from adopting a CBD methodology was the possibility to reuse components (Bass et al.

2000; Crnkovic and Larsson 2002; Ravichandran and Rothenberger 2003; Vitharana 2003).

However, empirical research on CBD has challenged these benefits and shown that 'it often took longer to develop a reusable component than to develop a system for a one-off purpose' (Huang et al. 2003). It was argued that the benefits of reuse are difficult to achieve in the first place; and they cannot be achieved immediately, but only in the long run (Crnkovic and Larsson 2002). Reuse of components allows companies to improve the productivity and quality of products; however, it may take a long time to develop software components, before they can be reused in a number of products.

Moreover, it was expected that adoption of CBD would further facilitate globally distributed development of software products, as happened in industries such as aeronautics, automotive, electronics and computer hardware, where CB architectures have been successfully used for setting up globally distributed design and production. For example, in the computer industry, Dell products include components produced by different vendors in various locations. In the automotive industry, the design of a car and the manufacture of car components involve designers and component suppliers at various dispersed locations (Olin et al. 1999). Even a very large and complex product such as an aircraft could be developed from remote locations, as in the case of Boeing-Rocketdyne (Malhotra et al. 2001), Boeing 777 (Yenne 2002) and Airbus.

Within the software industry, it was suggested that components could be developed in parallel independently by teams located in the same building or at remote locations. It was argued that CBD enables each site to take ownership of particular components and work on them independently without much need for inter-site communication and coordination (Colbert et al. 2001; Repenning et al. 2001). Carmel (1999) argued that adoption of component technology and CBD

would facilitate globalisation in the software industry because components could be developed remotely with minimum coordination across dispersed locations:

The software technology itself will continue to have an impact on global dispersion. Software globalization and dispersion will continue because of continued changes in underlying software technology. The industry is moving slowly, though unevenly, to a paradigm of software components. Small software components will be built and sold like subassemblies to be put together and made into larger, more comprehensive packages. Each of these small components will easily connect to other components. **These characteristics will allow distant teams of software developers to develop software components with only minimal coordination with others.** One future scenario is that low-cost nations will build the components and sell them to software design centers in industrialized countries for assembly (Carmel 1999:22; my emphasis).

### 1.3 A GAP IN THE LITERATURE

Nowadays, Globally Distributed Component-Based Development (GD CBD) is expected to become a promising area, as increasing numbers of companies are setting up software development in a globally distributed environment and at the same time are adopting a CBD methodology. Thus, being an emerging area, the management practice of GD CBD is evolving primarily on an *ad hoc* basis. At this time there is a lack of coherent, theory-based approaches for managing GD CBD. This process of globalization and adoption of CBD methodology has introduced potential benefits as well as new challenges in the management of software projects.

Numerous potential benefits can be associated with GD CBD. First, such a practice creates an expectation that companies involved in GD CBD will enjoy traditional benefits related to global distribution, such as lower development costs and shorter time-to-market. Second, globalisation of CBD promises to solve problems associated with CBD, such as lack of skilled professionals. In this respect GD CBD opens an opportunity to employ software engineers with required

skills to work on a project from dispersed geographical locations. Lastly, there is an expectation that the adoption of CBD by globally distributed organisations may mitigate coordination problems associated with traditional (non CB) GDSD.

On the other hand, research on co-located CBD projects has reported difficulties associated with the management of CBD projects, such as lack of stable standards, lack of reusable components, and problems related to the granularity and generality of components (Vitharana 2003). In the light of these problems, achieving the true potential of CBD, which is mainly about reusing components, is rather challenging in the context of co-located CBD (Crnkovic and Larsson 2002). Globally distributed organizations may face the above-mentioned and additional challenges (caused by geographical, time-zone and cultural differences) when adopting the practice of CBD.

So far, researchers in the Information Systems (IS) field have studied only limited aspects of the phenomenon of GD CBD: some have focused on the impact of globalization on the management of traditional (non CB) software development projects, while others have focused on the management of CBD in co-located projects. Research on management of GD CBD projects that combine these two streams is just emerging and is still in its early stages. At the time of this writing, after an extensive literature study I have identified three reports on GD CBD projects: a project by IBM described in Carmel (1999); Skandia's project described by Alexandersen et al. (2003); and lessons learned from GD CBD at Cisco Systems described by Turnlund (2004). This research on GD CBD reported that extensive coordination between people working from dispersed locations is required to succeed in GD CBD (these three projects will be discussed in greater detail in Chapter 2).

At present, little is known about how to organise and manage GD CBD successfully. Therefore, *this thesis aims to advance knowledge of the management*

*of GD CBD projects, and to develop a structured (theory-based) approach to the management of such projects.*

#### **1.4 RESEARCH OBJECTIVES AND QUESTION**

This thesis aims to develop a comprehensive understanding of the *management of Globally Distributed Component-Based Development projects*. The main research question is: ***how do companies organise and manage Component-Based Development in a globally distributed environment to be successful?***

To answer this question, the following objectives are set and steps undertaken:

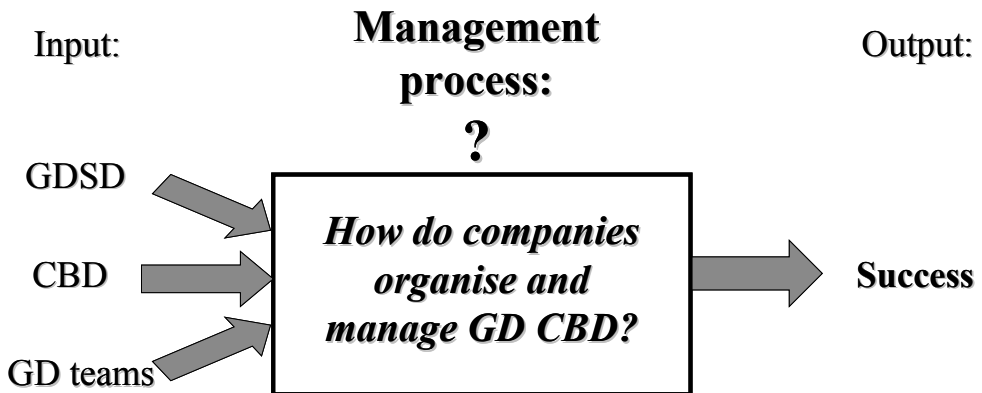
- First, to understand *what factors contribute to success in GD CBD projects*. A theoretical lens that indicates potential factors contributing to success in GD CBD is developed based on the existing literature; it serves as a starting point for the empirical investigation.
- Second, to collect *managerial practices that illustrate how companies organise and manage GD CBD projects*. Managerial practices are collected in four companies.

Based on the results of the empirical investigation, the theoretical lens is revised and a theoretical framework is proposed. Furthermore, managerial practices that describe how companies organise and manage CBD in globally distributed environment are presented.

## 1.5 FOCUS OF THIS RESEARCH

The focus of this research is on the management aspects of GD CBD projects, as described in Figure 1.

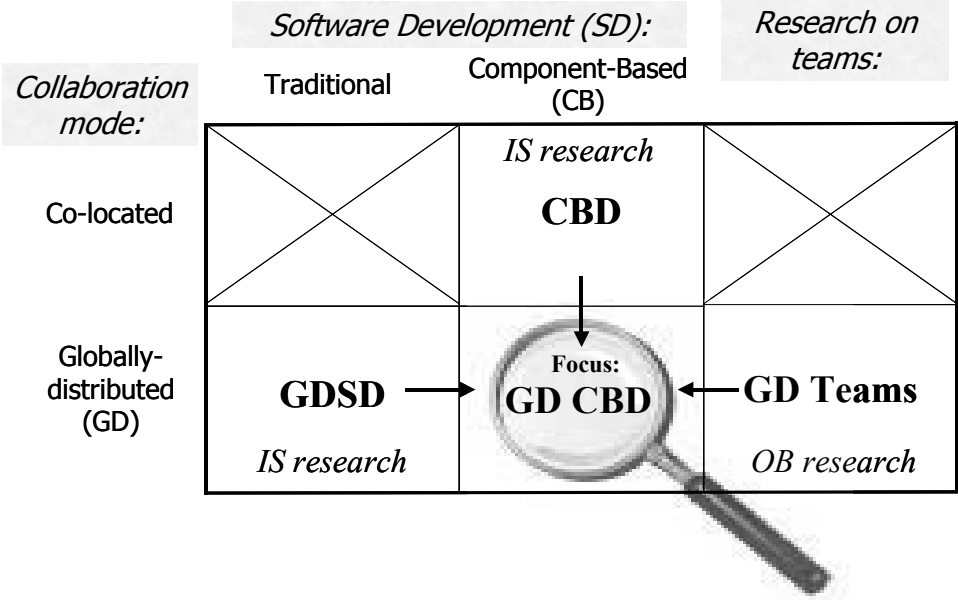
**Figure 1: Research focus**



A theoretical basis for studying the phenomenon of GD CBD draws upon the following related research streams (as illustrated in Figure 2): (i) research on traditional (non CB) GDSG which studies in depth the influence of global distribution on the management of software development projects; (ii) research on management of co-located and globally distributed CBD that discusses issues specific to CBD; and (iii) Organisational Behaviour (OB) research on collaboration in Globally Distributed (GD) teams that examines the importance of social aspects in global collaborations.



**Figure 2: Related research streams**



Different types of GD CBD can be distinguished: GD CBD for commercial purposes, component markets on the Internet, and Open Source Software development.

***Commercial GD CBD projects***

Commercial GD CBD projects are projects that develop software for commercial purposes, i.e. for specific customers or a large market of potential customers. GD CBD for commercial purposes can involve several sites of a multinational company (for example, in-house GD CBD by SAP and Microsoft that involve sites in USA, Europe and India), or it can involve several geographically dispersed companies that work together as a joint venture or based on an outsourcing agreement. ***This thesis focuses on commercial GD CBD projects that involve several sites of a multinational company.***

### ***Component Markets***

Software component markets allow bidding, buying and selling of components from geographically dispersed locations over the Internet. They have been suggested to be the most effective way to gain the benefits of components reuse (Szyperski 1998; Traas and van Hillegersberg 2000), and the most appropriate marketplace to buy and sell components is the Internet: ‘an international, freely accessible network which is perfectly suited for offering, promoting and distributing components’ (Traas and van Hillegersberg 2000:114). Producers of components and intermediary organizations offering components for sale are globally distributed. The Internet allows users to link these globally distributed entities on one web-site.

Software component markets on the Internet represent a globally distributed trade (buying and selling) of individual components, and not of projects of development of CB systems. Thus, component markets on the Internet are considered as a possible source (supplier) of components for projects developing CB systems, but not as an instance of such projects.

### ***Open Source Software Development***

Open Source Software (OSS) ‘is a software whose source code is distributed without charge or limitations on possible modifications and distributions by third parties’ (Crowston and Scozzi 2002:3). OSS has emerged from the hacker community (Wang and Wang 2001) facilitated by the Internet and the Web (Murugesan 1999). As stated on the Open Source community Web-site<sup>1</sup>:

The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs.

---

<sup>1</sup> [www.opensource.org](http://www.opensource.org)

After the success of several OSS projects such as Linux and Apache, the interest of the academic and commercial worlds in OSS has grown (Murugesan 1999; Crowston and Scozzi 2002). Development of OSS is entirely global and in most cases component-based: programmers from dispersed locations contribute to OSS via the Internet. Many OSS projects use component technologies, e.g. JavaBeans. However, as opposed to commercial GD CBD projects that are driven by business goals and involve paid staff, participation in development of OSS is voluntary (Murugesan 1999), and developers contribute to OSS ‘for the sake of peer recognition and personal satisfaction’ (Crowston and Scozzi 2002:3). Therefore, despite the fact that OSS is indeed an example of successful GD CBD, OSS and commercial GD CBD projects are different in their nature, and thus they need to be considered separately.

## **1.6 RELEVANCE AND CONTRIBUTIONS OF THIS RESEARCH**

This research is of relevance to IS research and to management practice. It provides the following theoretical and practical contributions.

### ***Theoretical contribution***

The research presented in this thesis has three main theoretical contributions. First, it offers a theoretical framework that identifies factors contributing to success in GD CBD is proposed.

Second, managerial practices that describe how companies successfully organise and manage CBD in a GD environment are offered. The framework and the managerial practices suggest a more structured (theory-based) approach to studying the management of GD CBD projects than the current research tradition.

Third, within the IS field, this thesis provides an integrated view on the phenomenon of GD CBD which combines three areas of research: (i) IS research on *traditional* (non-CB) GDSD projects, (ii) IS research on the management of *co-located* CBD, and (iii) OB research on collaboration in GD teams.

### ***Practical contribution***

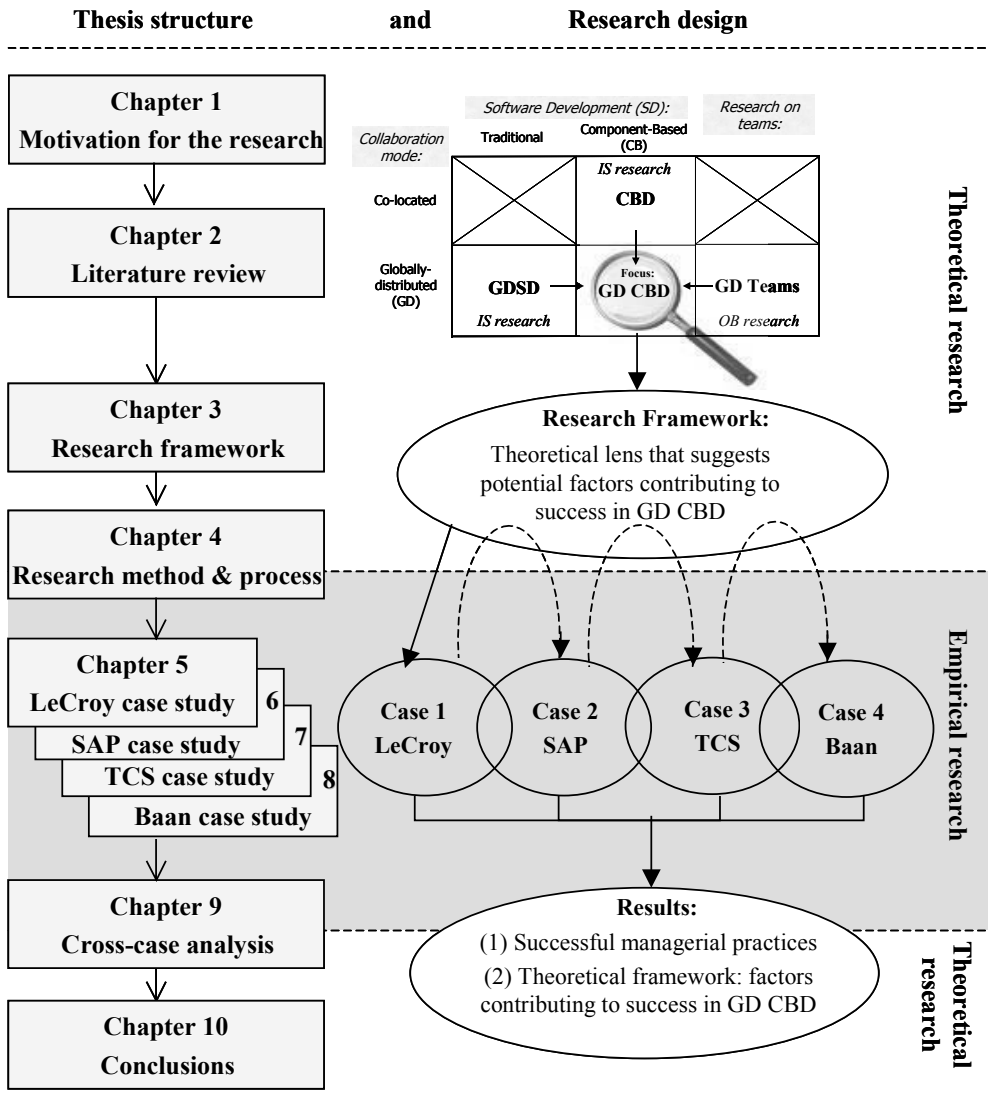
The research presented in this thesis is of much relevance to management practice. First, managerial practices perceived as contributing to success in the studied GD CBD projects are of high relevance to managers. Other companies involved in GD CBD could learn from these practices how to organise and manage GD CBD in their organizations.

Second, a checklist for managers is proposed. It identifies specific activities that help to implement the above-mentioned managerial practices in the management of actual GD CBD projects.

### **1.7 OUTLINE OF THIS THESIS**

The structure of this thesis is presented in Figure 3 below. Chapter 2 provides a review of the literature relevant to GD CBD. It covers the management of GD and co-located CBD, the management of traditional (non-CB) GSD projects, and collaboration in GD teams. Based on this literature a research framework is developed. The research framework, presented in Chapter 3, suggests potential factors contributing to success in GD CBD; it serves as a theoretical lens for empirical investigation. Chapter 4 explains the choice of qualitative case study methodology adopted in this research, and the case selection criteria. It also describes the research process and specific techniques used for data collection and analysis. Then, in Chapters 5-8, data analysis and results of four case studies at LeCroy, SAP, TCS and Baan (respectively) are presented and discussed. The cross-case analysis in Chapter 9 provides a comparison across cases and explains similarities and differences. Chapter 10 concludes the thesis by presenting the theoretical framework that identifies factors contributing to success in GD CBD, and outlining the contributions to research and practice, the possible limitations to the research, and suggestions for future research.

Figure 3: Thesis structure and research design



## CHAPTER 2 LITERATURE REVIEW

### 2.1 INTRODUCTION

This chapter gives an overview of the current research related to different aspects of the management of GD CBD projects. The findings and results reported in the relevant research streams are discussed and applied to the management of GD CBD projects.

Research on GD CBD is just emerging and is yet very limited. As discussed in Chapter 1, the three most closely related research areas that could provide an insight into the phenomenon of GD CBD are: research on GSD which studies in depth the influence of global distribution on the management of software development projects, research on co-located CBD, and OB research on GD teams which addresses the importance of human and social issues for success in globally distributed teams and alliances.

The main literature sources used to find relevant literature included online databases, such as ProQuest and the ACM digital library. Furthermore, all issues of major IS, OB and management journals from the last six years (1998-2004) were studied, including:

- *IS journals*: MIS Quarterly, IEEE Transactions on Engineering Management, IEEE Transactions on Software Engineering, Communications of the ACM, Information Systems Research, Journal of Information Technology, European Journal of IS and Journal of Management IS.
- *Management (general) journals*: Management Science, Academy of Management Journal, Academy of Management Review, Sloan Management Review and Harvard Business Review
- *OB journals*: Organization Science and Organization Studies

Searching for relevant literature, the main goal was to find *theoretical* and *empirical* papers that focused on GDSD, management of CBD (co-located and globally distributed) and collaboration in GD teams (the various social aspects involved in team collaboration). Within empirical papers the focus was on papers which were based on *actual* globally distributed projects and teams (rather than studies based on *students*).

This literature review is organised in the following way. Section 2.2 gives an overview of IS literature on the management of traditional (non-CB) GDSD projects. Following this, Section 2.3 focuses on collaboration in GD teams based on OB literature. Section 2.4 gives an overview of the management of CBD, globally distributed and co-located. Section 2.5 discusses how success is measured in IS projects. This chapter concludes with a summary of potential factors that may contribute to success in GD CBD (Section 2.6).

## **2.2 MANAGEMENT OF GLOBALLY DISTRIBUTED SOFTWARE DEVELOPMENT**

***Globally distributed software development projects*** are projects that consist of two or more teams working together to accomplish project goals from different geographical locations. In addition to geographical dispersion, globally distributed teams face time-zone and cultural differences that may include but are not limited to different language, national traditions, values and norms of behaviour (Carmel 1999).

Research on the management of GDSD started to emerge in the second half of the 1990s as a subset of research on management of globally distributed projects and, by the late 90s (e.g. Carmel 1999; Karolak, 1999), established itself as a separate research area. Historically, it has focused on traditional (non CB) software development.

Despite growing experience in the area of GDSD, research on this topic is still limited. Existing research is very fragmented and focuses on different aspects of distributed collaboration and at varying levels of analysis. For example, some studies focused on the various stages in GDSD, such as requirements analysis (Crowston and Kammerer 1998; Damian 2002; Damian et al. 2003), while others considered issues related to outsourcing software development to specific countries (Kumar and Willcocks 1996; Carmel 2003a, 2003b). Also, the unit of analysis used in IS development research varies among these studies: some studies considered the *organization* as the unit of analysis (e.g. Grinter et al. 1999; Kobitzsch et al. 2001; Orlikowski 2002); others focused on globally distributed software development *projects*, as does the research presented in this thesis.

IS literature that focuses on management of GDSD *projects* is examined in depth to develop a theoretical lens for studying GD CBD projects. This literature reported problems and breakdowns encountered in GDSD projects, and practices that helped to overcome these difficulties (e.g. Carmel 1999; Karolak 1999; Smith and Blanck 2002). Table 1 gives an overview of the core literature on management of GDSD projects, the main topics addressed in this literature, the research approaches applied and the data sources (marked as ‘V’).



**Table 1: Overview of core literature on management of GDSD projects**

	(Kotlarsky and Oshri 2005)
	(Sarker and Sahay 2004)
	(Herbsleb and Mockus 2003)
	(Espinosa and Carmel 2003)
	(Van Fenema 2002)
	(Orlikowski 2002)
	(Mockus and Herbsleb 2002)
	(Carmel and Agarwal 2002)
	(Smith and Blanck 2002)
	(Handel and Herbsleb 2002)
	(Kobitzsch et al. 2001)
	(Battin et al. 2001)
	(Ebert and De Neve 2001)
	(Herbsleb et al. 2000)
	(Carmel 1999)
	(Karolak 1999)
	(Herbsleb and Grinter 1999)
	(Grinter et al. 1999)
	(Grinter 1999)
	(Grinter 1998)
	(Jarvenpaa and Leidner 1998)
	(Grinter 1995)
Research approach:	
Qualitative case study	V
General overview	
Quantitative survey	V
Data source:	
Actual GDSD projects	V
Students	V
Main issues discussed:	
Problems & breakdowns	V
Coordination & control	V
Division of work	V
Communication patterns	V
Collaborative tools	V
Software engineering tools	
Inter-personal relationships	V
Knowledge sharing/ mgmt	V

As Table 1 shows, the vast majority of the core literature is based on the research conducted in *actual* GDSO projects; only two studies are based on the student teams. Qualitative case study methodology is the most popular research approach applied in these studies. The main issues addressed in this literature include problems and breakdowns in GDSO projects, and different managerial practices suggested to overcome the problems imposed by global distribution. These practices focused on activities to improve inter-site coordination, such as strategies for division of work, coordination and control mechanisms, communication patterns. Furthermore, technical support by means of (generic) collaborative tools and software engineering tools is suggested. Finally, some papers address issues related to inter-personal relationships and knowledge sharing / management. The following two sections will elaborate on the main findings and results reported in the literature on GDSO projects: Section 2.2.1 will discuss problems and breakdowns reported in GDSO projects and Section 2.2.2 will elaborate on practices suggested in this literature to organise and manage GDSO.

### **2.2.1 PROBLEMS AND BREAKDOWNS IN GDSO**

Traditionally, past studies on management of GDSO tended to focus on issues pertaining to the geographical dispersal of work. Naturally, because of several constraints associated with globally distributed work, such as *distance*, *time-zone* and *cultural differences*, traditional coordination and control mechanisms tend to be less effective in global development projects (Rafii 1995; Carmel 1999; Karolak 1999; van Fenema and Kumar 2000; Espinosa and Carmel 2003; Herbsleb and Mockus 2003). *Distance* reduces the intensity of communications, in particular when people experience problems with media that cannot substitute for face-to-face communications. *Cultural differences* expressed in different languages, values, working and communication habits and implicit assumptions are believed to be embedded in the collective knowledge of a specific culture (Baumard 1999), and thus may cause misunderstanding and conflicts. *Time-zone*

*differences* reduce opportunities for real time collaboration, as response time increases considerably when working hours at remote locations do not overlap. Therefore, receiving an answer to a simple question may take far longer than in co-located projects because of time-zone differences.

The IS literature reports problems and breakdowns in different areas encountered in GSD projects, and practices that help to overcome these difficulties (e.g. (Carmel 1999; Karolak 1999; Smith and Blanck 2002). The main problems and breakdowns reported in GSD projects are:

- Breakdown of traditional coordination and control mechanisms (Carmel 1999; van Fenema 2002; Cheng et al. 2004);
- Loss of communication richness (Carmel 1999; van Fenema 2002);
- Lack of understanding of counterpart's context (Orlikowski 2002);
- Language barriers (different competency in language) (Sarker and Sahay 2003);
- Misunderstandings caused by cultural differences (different conversational styles, different subjective interpretations) (Battin et al. 2001; Olson and Olson 2004);
- Loss of team cohesion and motivation to collaborate: decreased morale and lack of trust (Jarvenpaa and Leidner 1998; Carmel 1999; Karolak 1999);
- Asymmetry in distribution of information among sites (Carmel 1999);
- Difficulty in collaborating due to different skills and training, expertise in different tools and technologies, mismatch in IT infrastructure (van Fenema 2002; Sarker and Sahay 2004);
- Lack of informal, inter-personal communications (Herbsleb and Grinter 1999; van Fenema 2002);
- Difficulties to work in different time zones (Karolak 1999; Kobitzsch et al. 2001);
- Delays in distributed collaborative work processes: unproductive waits for the other side to respond with clarification or feedback (caused by time zone

differences and different interpretations) (Jarvenpaa and Leidner 1998; Herbsleb et al. 2000).

The surveyed studies report on successful and unsuccessful experiences of companies engaged in GSD projects and recommend a number of practices that would help these companies to reduce some of these problems imposed by global distribution. These practices will be discussed in the following section.

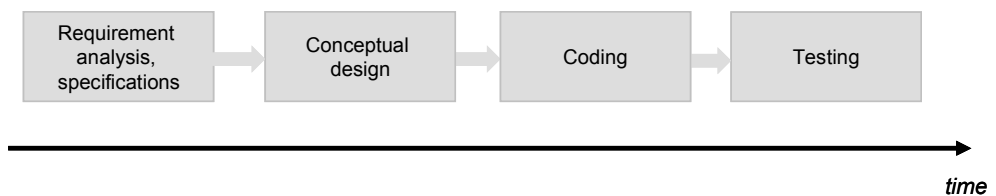
## 2.2.2 HOW TO ORGANISE AND MANAGE GSD

### 2.2.2.1 SOFTWARE DEVELOPMENT METHODOLOGY: BACKGROUND

Software development requires a multiplicity of tasks to be performed by multiple actors (Kraut and Streeler 1995). Typically, the structure of a software development process ‘relies on a guiding set of principles or methodology that defines roles, tasks, and their inter-relationships over time’ (Beath and Orlikowski 1994).

Historically, software systems have been developed following a Waterfall approach, which prescribes a number of phases to be followed in a sequential manner: requirements analysis and specifications, conceptual design, coding and testing, as illustrated in Figure 4.

**Figure 4: The Waterfall approach: traditional software development lifecycle**



The increasing complexity of current IS, combined with the demand for shorter project development cycles, has led to alternative approaches, such as parallel (concurrent) development based on organising tasks in an overlapping (parallel)

mode (Terwiesch and Loch 1996; van Fenema 2002). For contract-driven software (developed for a specific customer, as opposed to off-the-shelf, sometimes customisable, software systems), methodologies aiming to increase user participation in the development process have been developed (Hirschheim and Klein 1994). Examples of these methodologies include Joint Application Development (JAD), Rapid Application Development (RAD) and prototyping (Carmel et al. 1993; Trevor 1994; Beynon-Davies et al. 1999; van Fenema 2002).

The choice of a development methodology (e.g. Waterfall and/or parallel) has implications for the way development work is divided and integrated (Kraut and Streeler 1995; van Fenema 2002). In particular, organising sequentially dependent tasks to be conducted in parallel changes the way tasks are coordinated and controlled. For example, overlap between tasks means that developers cannot check output from preceding tasks and compare them to standards, and therefore they need rely on inter-personal communications (van Fenema 2002). In a globally distributed environment work is divided between teams and individuals at multiple geographically dispersed locations, and thus coordination and integration of work need to be done across these remote locations.

As discussed in the previous section, GDSD projects suffer from coordination breakdowns (Carmel 1999; Karolak 1999). The literature on GDSD has suggested a number of practices that would help to organise and manage GDSD projects to overcome (potential) problems and breakdowns (these practices are discussed in the following two sections). As Table 1 illustrated, these practices mainly focus on *inter-site coordination*, aiming to improve coordination between remote sites, and *tools and technologies* that make it possible to collaborate in a distributed mode.

#### **2.2.2.2 INTER-SITE COORDINATION IN GDSD PROJECTS**

Managerial practices for inter-site coordination in GDSD suggested in the literature involve (i) strategies regarding *division of work*, which aim to make

easier coordination and integration of work conducted at remote locations, (ii) specific *coordination mechanisms* adapted for a distributed environment, and (iii) *communication patterns* aiming to make inter-site coordination more efficient, through planning systematic communication between remote counterparts and establishing rules of communications. These three groups of practices are now discussed below.

---

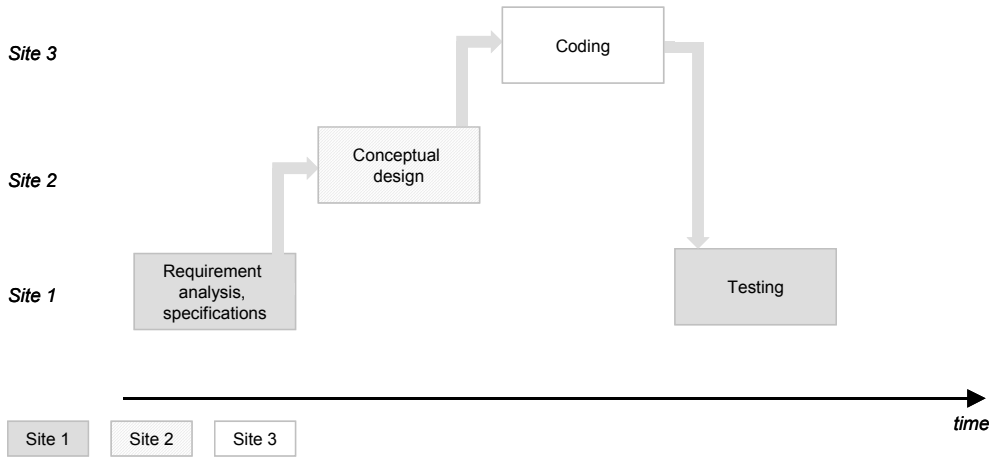
### **i. Division of work**

---

In GDSD, work-packages assigned to remote sites need to be managed and coordinated to ensure a successful outcome. Typically, strategies to divide work between locations suggested in the IS literature on management of GDSD projects aim to reduce needs for inter-site coordination and communications. In particular, strategies recommended for division of work are:

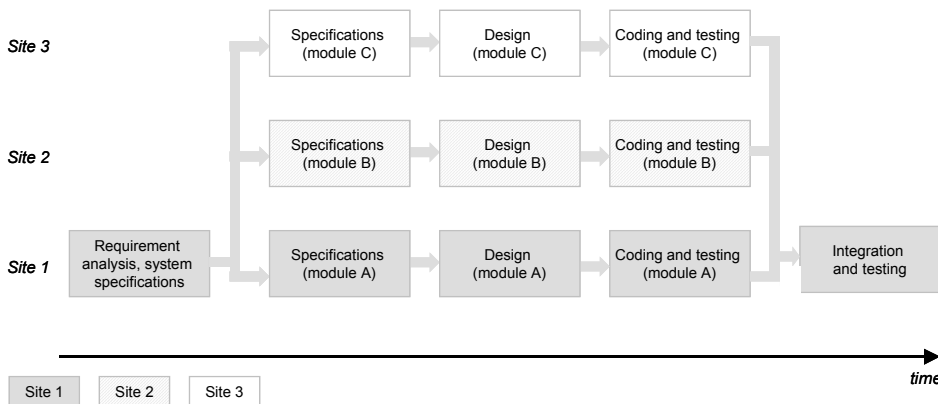
- Division of work by *phase / process step* when globally dispersed sites engage in different phases of a project in a sequential manner (i.e. work is handed over to a remote site after completing certain process steps) (Carmel 1999; Grinter et al. 1999), as illustrated in Figure 5 (Scenario A).

**Figure 5: Scenario A - GSDS organised by phase / process step**



- Division of work by *product structure (product module)* when each product module / feature is developed at a single site (Carmel 1999; Grinter et al. 1999). This approach allows for different sites to work on different modules in parallel. Figure 6 illustrates a possible scenario (Scenario B), when a system is divided into modules (typically different product functions) and each module is allocated to a different site.

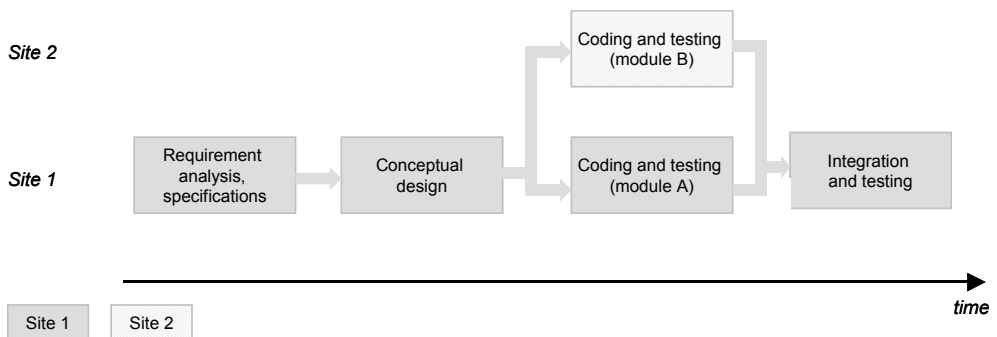
**Figure 6: Scenario B - GSDS organised by product structure (product module)**



Structured, well-defined tasks are more suitable to be allocated by *phase / process step*, while abstract (unstructured, loosely defined) tasks are more appropriate to be allocated by *product structure (product module)* (Carmel 1999; Karolak 1999; van Fenema 2002).

- Division of work that *minimizes requirements for cross-site communication and synchronization* in the context of particular types of product architecture and mechanisms for coordinating work (Ebert and De Neve 2001; Reppenning et al. 2001; Herbsleb and Mockus 2003). To achieve this, it was recommended that ‘tightly coupled work items that require frequent coordination and synchronization should be performed within one site’ (Mockus and Weiss 2001). Figure 7 illustrates a possible scenario (Scenario C) with tasks that require frequent coordination: requirements analysis and specification, conceptual design and integration and testing are conducted at one site, and only well defined tasks (coding and testing of different modules) are conducted at two locations in parallel.

**Figure 7: Scenario C – only well-defined tasks distributed across locations**

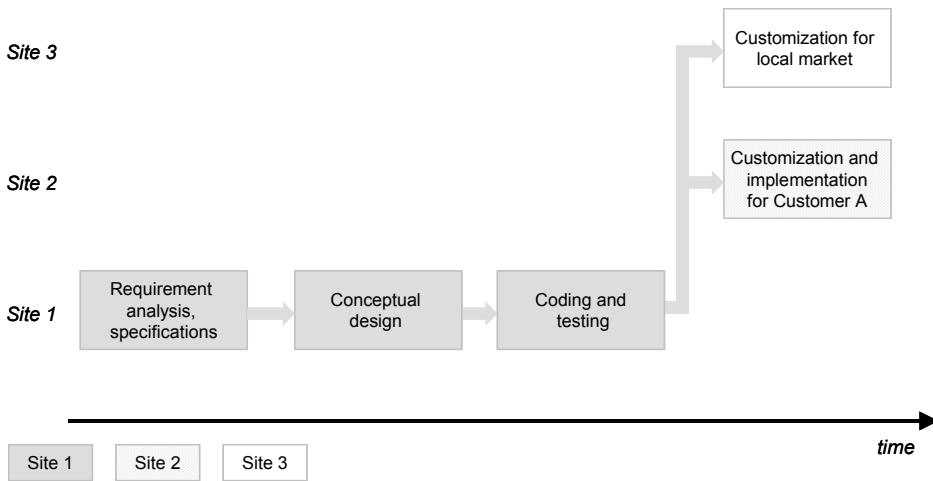


- Division of work based on *product customisation*, so that one site develops the product and other sites perform customisation, i.e. changes such as adding features and enhancements for specific customers (Grinter et al. 1999). In this



case, often sites that customise the product are in the proximity of a customer. Figure 8 illustrates a possible scenario (Scenario D) when a system is developed in one location (site 1), and other globally dispersed sites customise the system for specific customers (site 2) or local markets, i.e. large number of potential (local) customers (site 3).

**Figure 8: Scenario C – GSD based on *product customisation***



## ii. Coordination mechanisms

A number of coordination mechanisms are recommended to overcome problems and breakdowns in GSD projects. These are, in particular:

- A more explicit, documented and formalized project process: standardizing and documenting the development methodology, distributing it across sites and storing it in a shared repository; educating all team members in the chosen methodologies (Carmel 1999; Karolak 1999; Mockus and Herbsleb 2002; van Fenema 2002).
- Encouraging visits to remote sites and face-to-face meetings (Carmel 1999; Smith and Blanck 2002; van Fenema 2002)

- Establishing liaisons between remote locations (Battin et al. 2001; van Fenema 2002)
- Rotating team members (Carmel 1999; Carmel and Agarwal 2001; Smith and Blanck 2002)
- Creating transparency in project goals and company vision (Carmel 1999; Ebert and De Neve 2001)
- Building awareness of the work conducted at remote sites (e.g. making project plans accessible over the Web); of remote teams (e.g. creating a web-page for each team member with personal information); and of local context (e.g. providing information about local working hours and holidays) (Kobylinski et al. 2002; Mockus and Herbsleb 2002; Smith and Blanck 2002; Espinosa et al. 2003).

### **iii. Communication patterns**

Communication patterns recommended for GDSD teams include the following:

- Scheduling systematic phone/video meetings between remote counterparts (managers and team members) (Karolak 1999; Herbsleb and Mockus 2003).
- Establishing communication protocols that cover ground rules and expectations concerning communications (e.g. for emails) (Carmel 1999; Olson and Olson 2004; Sarker and Sahay 2004).
- Communicating laterally (Carmel 1999; Smith and Blanck 2002; van Fenema 2002; Herbsleb and Mockus 2003).
- Being clear and patient in communications, as counterparts might not be able to comprehend and communicate in English (Carmel 1999; Smith and Blanck 2002; Espinosa et al. 2003; Paasivaara 2003).
- Investing in language and cultural training (Battin et al. 2001; Smith and Blanck 2002).

### 2.2.2.3 TOOLS AND TECHNOLOGIES TO SUPPORT GDSD

Tools and technologies suggested to overcome problems and breakdowns in GDSD and enable collaboration in a distributed environment comprise (i) a powerful *ICT infrastructure* that allows the transfer of data at high speed, (ii) *generic collaborative technologies* enabling remote colleagues to connect and communicate, and (iii) *software engineering tools* that support software development activities conducted in parallel at remote locations.

#### **i. ICT infrastructure**

A reliable and high bandwidth ICT infrastructure is required to ensure connectivity between remote sites (Carmel 1999; van Fenema 2002).

#### **ii. Collaborative technology**

Collaborative technology can be used to improve collaboration in GDSD teams. The most commonly suggested collaborative technologies are:

<ul style="list-style-type: none"><li>• Email</li><li>• Chat (Instant Messaging)</li><li>• Phone / audio conference</li><li>• Videoconference</li><li>• Internet/intranet</li><li>• Group calendar</li><li>• Discussion list</li><li>• Electronic meeting system</li></ul>	<p><i>References:</i></p> <p>(Sarker and Sahay 2004) (Herbsleb and Mockus 2003) (Smith and Blanck 2002) (Herbsleb et al. 2002) (van Fenema 2002) (Carmel and Agarwal 2002) (Mockus and Herbsleb 2002) (Handel and Herbsleb 2002) (Ebert and De Neve 2001) (Battin et al. 2001) (Herbsleb et al. 2000) (Carmel 1999) (Karolak 1999)</p>
--	--

Typically, collaborative technologies recommended for GDSD teams are classified according to the time/space dimension: the two-by-two same/different place and

same/different time matrix proposed in computer mediated communications literature (DeSanctis and Gallupe 1987) was widely supported in the research on GDSD projects to classify collaborative technologies (e.g. Carmel 1999; Smith and Blanck 2002). This matrix contains four categories and corresponding technologies: *same place/ same time* (collocated group decision support), *same place/different time* (workflow systems), *same time/different place* (telephone, chatting), *different place/different time* (bulletin board). However, this framework does not take recent technical progress into account, e.g. mobile technology and advanced collaborative tools such as Groove (<http://www.groove.net>).

A more advanced classification of collaborative technologies was suggested by Huis et al. (2002): the authors distinguish between several types of collaborative technology that support different needs of globally distributed teams in different time/place settings, as illustrated in Table 2.

**Table 2: Types of Collaboration Technology (adopted from Huis et al. 2002)**

	<b>Setting</b>		
	<b>Different place/ different time (off-line)</b> , i.e. <i>support between encounters</i>	<b>Different place/ same time (on- line)</b> , i.e. <i>support for electronic encounters</i>	<b>Same place/ same time,</b> i.e. <i>support for face-to- face meetings</i>
<p><b>Communication Systems:</b> <i>aim to make communications between remote people easy, cheap and fast</i></p>	<ul style="list-style-type: none"> <li>• fax</li> <li>• email</li> <li>• voice-mail</li> <li>• video-mail</li> </ul>	<ul style="list-style-type: none"> <li>• telephone</li> <li>• mobile phone</li> <li>• desktop-video</li> <li>• video / audio-conferencing systems (multi-point)</li> <li>• chat system</li> </ul>	
<p><b>Information sharing systems:</b> <i>aim to make the storage and retrieval of large amounts of information quick, easy, reliable and inexpensive</i></p>	<ul style="list-style-type: none"> <li>• document sharing systems</li> <li>• computer conferencing</li> </ul>	<ul style="list-style-type: none"> <li>• tele-consultation systems</li> <li>• application for searching remote information sources</li> </ul>	<ul style="list-style-type: none"> <li>• presentation systems</li> </ul>
<p><b>Collaboration systems:</b> <i>aim to improve teamwork by providing document sharing and co-authoring facilities</i></p>	<ul style="list-style-type: none"> <li>• co-editing systems</li> </ul>	<ul style="list-style-type: none"> <li>• shared white-board, CAD, word-process or spread-sheet</li> </ul>	<ul style="list-style-type: none"> <li>• Group Decision Support Systems</li> </ul>
<p><b>Coordination systems:</b> <i>aim to coordinate distributed teamwork by coordinating work processes</i></p>	<p>Synchronizers:</p> <ul style="list-style-type: none"> <li>• group calendar</li> <li>• shared project planning</li> <li>• shared workflow system</li> </ul>	<ul style="list-style-type: none"> <li>• awareness / notification systems (e.g. 'active batch')</li> </ul>	<ul style="list-style-type: none"> <li>• command and control centre support systems</li> </ul>
<p><b>Social encounter systems:</b> <i>aim to facilitate unintended interactions</i></p>		<ul style="list-style-type: none"> <li>• media spaces</li> <li>• virtual spaces</li> </ul>	

### iii. Tools to support software engineering

In addition to collaborative technologies that are generic to a great extent, a number of tools specific to software development are suggested to support GDSD teams. The most commonly suggested tools include the following:

<ul style="list-style-type: none"><li>• Configuration and version management tool</li><li>• Source-management system</li><li>• Document management system</li><li>• Replicated databases / repositories</li><li>• CASE tools that support modelling and visibility of design</li><li>• Integrated Development Environment (IDE) toolset, which combines tools such as editor, compiler, debugger</li></ul>	<p><i>References:</i></p> <p>(Cheng et al. 2004) (Smith and Blanck 2002) (Carmel and Agarwal 2002) (Mockus and Herbsleb 2002) (Handel and Herbsleb 2002) (Ebert and De Neve 2001) (Battin et al. 2001) (Herbsleb et al. 2000) (Carmel 1999) (Karolak 1999) (Grinter 1999) (Grinter 1995)</p>
--	--

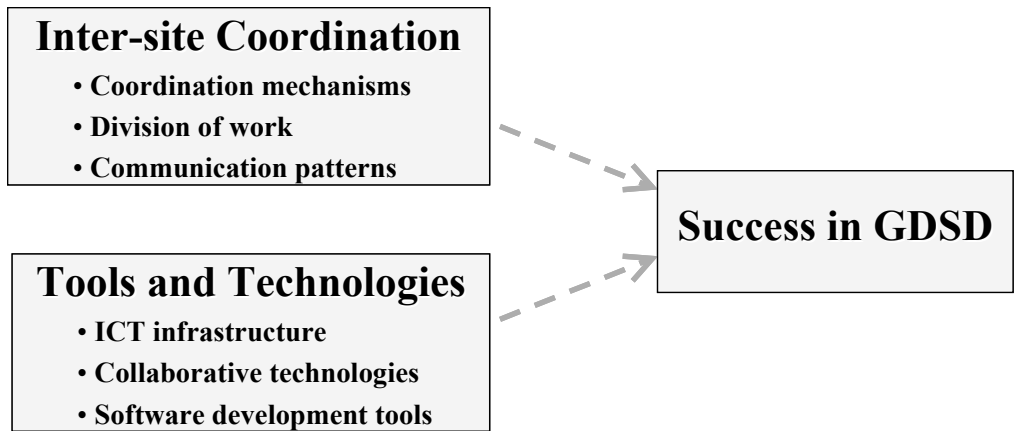
These tools ensure consistency in the product and development environment across dispersed locations.

Furthermore, adding collaborative capabilities such as email, Instant Messaging (IM), screen sharing, and a configuration management tool to the Integrated Development Environment (IDE) toolset was recommended by Cheng et al. (2004) to deal with breakdown in communication and coordination among developers. Cheng et al. (2004) argue that integrating collaborative capabilities into IDE holds great potential for easing programmers' development activities. However, this integration introduces a number of technical and design challenges, in particular: (i) building for extensibility, interoperability, and flexibility; (ii) choosing and designing the 'right' set of collaborative features; and (iii) supporting transitions between individual and group work.

### 2.2.3 CONCLUSIONS FROM IS RESEARCH ON THE MANAGEMENT OF GDSD PROJECTS

Past research in the IS field suggests that the proper application of technical and operational mechanisms, such as tools, technologies and coordination mechanisms, is the chief factor that may lead to successful GDSD projects (Carmel 1999; Karolak 1999; Herbsleb et al. 2002). Figure 9 illustrates schematically the potential factors that contribute to success in GDSD projects.

**Figure 9: Potential factors that contribute to success in GDSD projects.**



Overwhelmingly, the solutions (discussed in Sections 2.2.2.2 and 2.2.2.3) proposed to support globally distributed teams are technical in nature, involving little attention to the human and social aspects involved in globally distributed work (Karolak 1999; Battin et al. 2001; Ebert and De Neve 2001; Espinosa and Carmel 2003). Furthermore, in the few studies that focused on the social aspects of globally distributed work, trust and social communications were indicated as barriers to achieving success between globally distributed teams. For example, Jarvenpaa and Leidner (1998) indicated that a lack of trust is likely to develop between globally distributed teams, while Carmel (1999) raised a concern about

possible breakdowns in communications that may cause coordination problems because of language barriers, cultural differences, asymmetry in distribution of information among sites, and a lack of team spirit. In essence, past research is rather concerned with the barriers that social aspects present to globally distributed collaboration (Jarvenpaa et al. 1998; Jarvenpaa and Leidner 1998; Carmel 1999; Karolak 1999).

While, traditionally, the main focus of the IS literature on globally distributed projects has been on coordination and technical aspects related to the management of GDSD projects, OB research has acknowledged the importance of social-related aspects in global collaborations, such as trust and inter-personal relationships (Storck 2000; Child 2001). Furthermore, it is suggested that knowledge sharing is important for success in globally distributed teams (Faraj and Sproull 2000; Orlikowski 2002).

Section 2.3 will elaborate on the role of social aspects and knowledge sharing for success in GD teams, based on the IS literature and OB research.



## 2.3 RESEARCH ON COLLABORATION IN GLOBALLY DISTRIBUTED TEAMS<sup>2</sup>

### 2.3.1 SOCIAL ASPECTS IN GLOBALLY DISTRIBUTED TEAMS

Social aspects are studied in depth in OB research. Among the many social-related factors contributing to collaboration, past studies have considered formal and informal communications (Storck 2000; Child 2001; Dyer 2001), trust (Storck 2000; Arino et al. 2001; Ba 2001; Child 2001), motivation (Child 2001) and social ties (Granovetter 1973; Storck 2000; Child 2001). The literature on IS development projects is far more limited in addressing the impact that social-related factors may have on success in software development projects. As argued above, past studies related to software development in the context of globally distributed teams have mainly raised concerns about managers' ability to overcome geographical, time-zone and cultural differences. For example, according to Smith and Blanck (2002:294), 'an effective team depends on open, effective communication, which in turn depends on trust among members. Thus, trust is the foundation, but it is also the very quality that is most difficult to build at a distance'.

**Trust** is defined by Child (2001:275) as 'the willingness of the one person or group to relate to another in the belief that the other's action will be beneficial rather than detrimental, even though this cannot be guaranteed'.

---

<sup>2</sup> Material presented in section 2.3 is published in a paper by Kotlarsky and Oshri (2005) 'Social ties, knowledge sharing and successful collaboration in globally distributed system development projects'. *European Journal of Information Systems* 14(1) pp.37-48.

Trust is more likely to be built if personal contact, frequent interactions and socializing between teams and individuals are facilitated (Arino et al. 2001; Ba 2001; Child 2001).

In addition, rapport, which was identified as important for collaboration between project teams and individuals, is more likely to be fostered in a co-located environment (Gremler and Gwinner 2000).

**Rapport** is defined as ‘the quality of the relation or connection between interactants, marked by harmony, conformity, accord, and affinity’ (Bernieri et al. 1994).

It has also been argued that informal communications play a critical role in coordination activities leading to success in co-located software development (Kraut and Streeler 1995). As the size and complexity of IS development increase, the need for supporting informal communications also increases dramatically (Herbsleb and Moitra 2001). Consequently, in distributed development projects the amount of such communication is greatly reduced as a result of time, cultural differences and geographical distance (Grinter et al. 1999): this, in turn, leads to the majority of problems reported in GDSD projects (discussed in Section 2.2.1 ). For example, lack of interpersonal communications between remote team members and limited mutual knowledge are argued to be factors contributing to breakdowns in coordination and communication (Crampton 2001; Hansen 2002; Orlikowski 2002).

Furthermore, a related study on distributed social networks by Herbsleb and Mockus (2003) suggested that distributed social networks may be less effective

than local social networks<sup>3</sup>. Their research reveals that (i) distributed social networks are much smaller than same-site social networks, (ii) there is far less frequent communication in distributed social networks compared to same-site social networks, (iii) people find it much more difficult to identify distant colleagues with necessary expertise and to communicate effectively with them, and (iv) people at different sites are less likely to perceive themselves as part of the same team than people who are at the same site.

By and large, studies in the IS field tend to treat the social aspects involved in globally distributed IS development projects as constraints, while OB research offers evidence suggesting that factors such as trust and rapport have a positive impact on global collaboration (Storck 2000; Child 2001).

### **2.3.2 KNOWLEDGE SHARING IN GLOBALLY DISTRIBUTED TEAMS**

Past studies on knowledge sharing in IS development projects have focused mainly on co-located sites (Faraj and Sproull 2000; Massey et al. 2002), whereas the discussion of knowledge-sharing mechanisms and the contribution of knowledge-sharing activities to success in the context of distributed IS teams is still limited. Existing studies on GDSD have reported on the problems caused by lack of shared knowledge (Kobitzsch et al. 2001; Herbsleb and Mockus 2003).

However, organisational behaviour studies on GD teams have recognised the importance of knowledge sharing for the success of such teams (Majchrzak et al. 2000; Storck 2000; Crampton 2001; Hansen 2002; Orlikowski 2002). For

---

<sup>3</sup> Herbsleb and Mockus (2003) define *social network* as a network of people with whom one interacts with a frequency that varies from more than once a day to a few times a year. A *distributed social network* is a social network that involves people from dispersed locations, while a *same-site social network* is a social network that involves people from one location.

example, Storck (2000) claims that sharing knowledge is important to building trust and improving the effectiveness of group work. Herbsleb and Moitra (2001) reiterate this observation, claiming that without an effective sharing of information, projects might suffer from coordination problems leading to unsuccessful project outcomes.

Other studies have described the complexity involved in sharing knowledge in co-located sites. For example, it has been acknowledged that the sharing of knowledge is a rather difficult task because of the idea that knowledge can be tacit (Polanyi 1967). The knowledge transformation model proposed by Nonaka and Takeuchi (1995), who conducted their research in co-located sites of electronic goods companies in Japan, is one example that demonstrates the complexity involved in transforming tacit to explicit knowledge and vice versa.

Additional support to the above view is provided by Faraj and Sproull (2000), who claim that instead of sharing specialized knowledge individuals should focus on knowing where expertise is located and needed. Such an approach towards knowledge sharing is also known as *transactive memory* (Wegner 1987).

***Transactive memory*** is defined as the set of knowledge possessed by group members coupled with an awareness of who knows what (Wegner 1987).

Transactive memory was found to positively affect group performance and collaboration by quickly bringing the needed expertise to knowledge seekers (Faraj and Sproull 2000; Storck 2000). The transactive memory of a globally distributed team implies that team members know the composition of a remote team (the people and their roles) and know the areas of expertise of their remote counterparts.

Further implications for knowledge sharing may arise when teams are faced with cultural, geographical and time-zone differences in globally distributed work.

Herbsleb, Mockus et al. (2000:3) described in their study how one global IS development project was facing major challenges to identify who knows what:

There was a nearly total absence of informal, unplanned communications across sites. The difficulties of knowing who to contact about what, of initiating contact, and of communicating effectively across sites, led to a number of serious coordination problems.

Similarly, a need to know whom to contact about what was reported in the studies of Grant (1996), Herbsleb and Grinter (1999), Orlikowski (2002), and Herbsleb and Mockus (2003).

Indeed, research has suggested that such hurdles in managing distributed projects could be avoided through the build-up of *collective knowledge* (also referred as *common knowledge*), which comprises elements of knowledge that are common to all members of an organisation (Grant 1996). In the case of GD CBD projects, the ‘organisation’ involves all people participating in the globally distributed project from their remote locations.

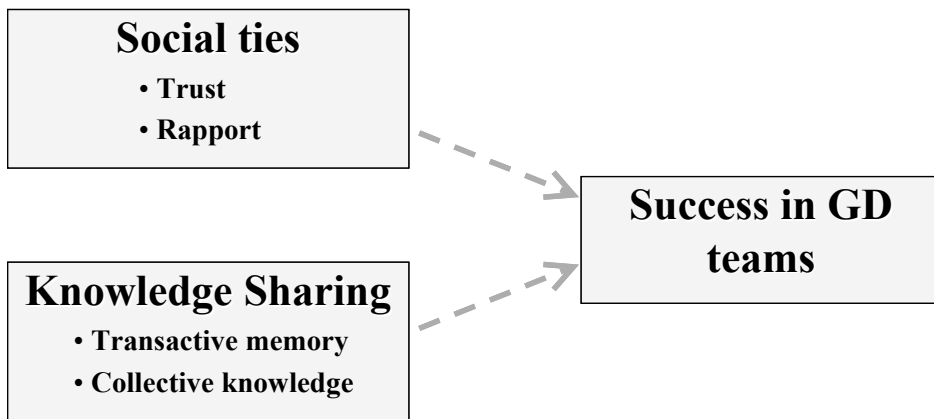
***Collective knowledge*** comprises the profound knowledge of an environment, of established rules, laws and regulations and ‘knowledge of an unspoken, of the invisible structure of a situation, a certain wisdom’ (Baumard 1999). It includes language, other forms of symbolic communication and shared meaning (Grant 1996).

*Collective knowledge* is based on the wisdom of social experience (Baumard 1999). In co-located organizations this would mean the development of a collective mind (Weick and Roberts 1993; Crowston and Kammerer 1998; Weick et al. 1999) through participation in tasks and social rituals (Orr 1990; Orlikowski 2002).

### 2.3.3 CONCLUSIONS FROM OB RESEARCH ON GLOBALLY DISTRIBUTED TEAMS

The OB literature offers several factors, such as trust, rapport, transactive memory and collective knowledge, that may positively affect collaboration through social activities and personal interactions. Figure 10 illustrates schematically the potential factors that may contribute to success in GD teams.

**Figure 10: Potential factors that contribute to success in GD teams**



The literature discussed above is largely focused on traditional (non CB) GDSD (Section 2.2) and gives an OB research perspective on GD teams (Section 2.3). The following Section 2.4 will focus on CBD. It will explain the background of CBD and related concepts, and will give an overview of CBD in globally distributed and co-located environments, based on the IS literature.

## 2.4 MANAGEMENT OF COMPONENT-BASED SOFTWARE DEVELOPMENT

*The whole idea is that we can take the bunch of different components and create a different instrument within weeks is kind of optimistic, but within a few months rather than in a few years.*

(Larry Salant, Director of Software Engineering, LeCroy)

### 2.4.1 A HISTORICAL PERSPECTIVE ON COMPONENT-BASED DEVELOPMENT

CBD has its roots in manufacturing. The trend to develop products that have a *Component-Based* or *modular* architecture is well established in automotive, electronics, aircraft and other manufacturing types of industries<sup>4</sup>. Since the mid-1960s, when the concept of modular production (Starr 1965) was introduced, modular (later referred to as component-based) product architectures became dominant in manufacturing industries.

In manufacturing, **components** (or *modules*)<sup>5</sup> are defined as parts of an assembly, chunks that ‘implement one or a few functional elements in their entirety’ (Ulrich and Eppinger 2000).

A **Component-Based system** is a system that has two properties: (1) components, and (2) connections (interactions) between components that are well defined and are generally fundamental to the primary functions of the product (Ulrich and Eppinger 2000).

<sup>4</sup> In manufacturing, a product is assembled from parts, as opposed to process industries, where a production process is based on mixing raw materials and/or chemical processes (e.g. chemical, pharmaceutical, food industries).

<sup>5</sup> There is some confusion among practitioners as well as academics regarding the definition of a *component* and a *module*. Typically, *components* imply finer granularity than *modules* (a module could consist of a number of components). However, in practice and academic literature these two terms are often used interchangeably.

As opposed to a monolithic system, a CB system potentially has a number of advantages for production, and can increase the competitive advantage of a company in the market.

First, a CB system allows changes to be made to isolated functional elements of the product without affecting the design of other components (Ulrich and Eppinger 2000). Thus, changes in a product could be made fairly easily and quickly (as changes in different components could be done in parallel and without causing unwanted side-effects).

Second, from a marketing perspective, having a CB system enables easier customisation by facilitating different product configurations for different users and different markets (e.g. the same car model designed for different countries can be somewhat different), and increases product variety (the range of product models). In particular, a CB system architecture (structure) makes the integration of components easier, which is important for:

- (i) upgrades (the possibility to replace a component, typically by a more recent version), as technological capabilities or users' needs evolve;
- (ii) add-ons (adding components by a third party) according to a user's needs; and
- (iii) flexibility in use, as some products can be configured by users to provide different capabilities (e.g. many cameras can be used with a different lens and flash options).

In each of these cases, a CB architecture allows a minimization of the physical changes required to achieve a functional change (Ulrich and Eppinger 2000). In a CB system components could be integrated relatively easily either by a *vendor* or by an *end user*:

- *Vendor integration*: Dell is an example of a vendor that assembles computers from pre-defined components, according to the specific choice of a customer.
- *End-user integration*: Many products are sold by a manufacturer as a basic unit, to which users can add components, often produced by third parties,



according to their specific needs. For instance, the computer is a basic unit to which third-party storage devices (e.g. CD-RW, memory key, zip drive) could be added according to customer needs and personal preferences.

Third, standardization of components allows the use of the same component in multiple products, thus reducing time-to-market, and production costs (Ulrich and Eppinger 2000). Time-to-market is shorter because reusing components saves the time required for design and quality assurance of these components. Production costs are lower, because fixed costs for setting up production lines and equipment are divided over more components as batch size increases. Similarly, suppliers often give a quantity discount for larger quantities of components to be procured. Moreover, knowledge and experience invested in the design of a component that is later reused in a number of products implies reuse of this knowledge and experience.

Adopting CB design facilitated globalisation in manufacturing industries, because a CB system is relatively easy to develop from dispersed locations and/or it is possible to buy parts from suppliers located all over the world. For instance, in the computer industry, Dell products include components produced by different vendors in various locations. In the automotive industry, the design of a car and the building of car components involves designers and component suppliers at various dispersed locations (Olin et al. 1999). Even a very large and complex product such as an aircraft could be developed from remote locations, as in the case of the Boeing Rocketdyne (Malhotra et al. 2001), Boeing 777 (Yenne 2002) and Airbus.

In the light of CB systems, a number of similarities could be observed between the manufacturing and software worlds. Similar to manufacturing, in the software world a system can be integrated from components by a vendor or an end-user. For example, an end-user can buy separately Internet Explorer, Adobe Acrobat and

Microsoft Office and plug (install) them together. In addition, vendors (e.g. SAP, PeopleSoft) of complex software systems, such as ERP (Enterprise Resource Planning), SCM (Supply Chain Management) and CRM (Customer Relationship Management) systems, typically integrate components required by a customer for that customer.

The next section will elaborate on CBD trends in the software industry.

#### **2.4.2 CBD IN THE SOFTWARE INDUSTRY: BACKGROUND**

In the software development industry CBD is a relatively new trend. CBD emerged in the mid-90s with the introduction of *software component technologies* such as Enterprise JavaBeans, Microsoft COM and CORBA, and is increasingly becoming a major trend in software development.

***Software component technology*** includes the software that provides a runtime environment for software components (sometimes called a component *framework*), as well as other tools useful for designing, building, combining, or deploying components or applications built from components (Bass et al. 2000).

Information Technology (IT) providers are turning to software component technologies as the most promising way of meeting demands for increased productivity, reduced time-to-market and improved system quality (Peters and Pedrycz 2000; Kim 2002).

***Component-Based (Software) Development*** involves (i) development of software components and (ii) building software systems through the planned integration of pre-existing (developed in-house or procured from the component market) software components (Bass et al. 2000). CBD also involves reusing application frameworks, which provide the architecture for assembling components into a software system (Vitharana 2003).

**CBD Methodology** comprises (i) software component technology and technical steps for (ii) designing and implementing software components, (iii) assembling systems from pre-built software components, and (iv) deploying assembled systems in their target environment (Bass et al. 2000).

As a result of applying CBD methodology, a *Component-Based system* is developed.

On a conceptual level, a **component-based system** could be described as consisting of components that are integrated by means of interfaces (similar to a CB system in manufacturing, as described in Section 2.4.1). On a more detailed level, there are numerous ways to design and integrate components.

The next section will elaborate on the definitions and main characteristics of software components, and explain how components can be integrated into a CB system.

### **2.4.3 A COMPONENT-BASED SYSTEM**

#### **2.4.3.1 WHAT IS A SOFTWARE COMPONENT?**

Similarly to manufacturing, in software development there is some confusion about what a *component* is:

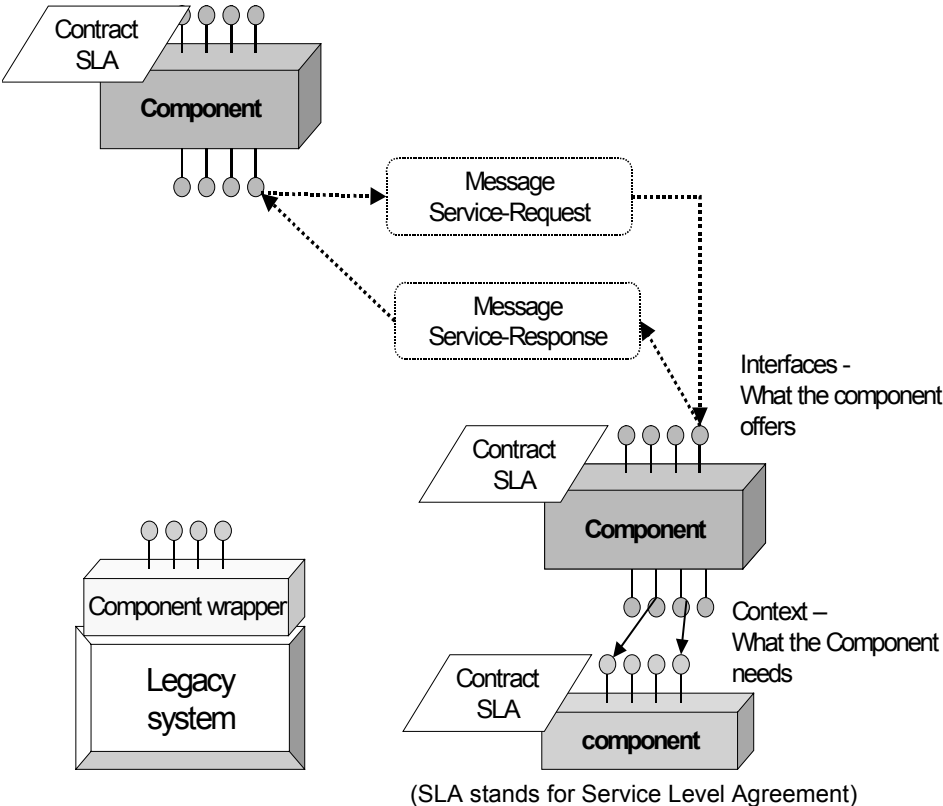
Industry doesn't speak consistently to the question of what a software component is. Some equate commercial-off-the-shelf (COTS) software packages with components. Some consider the use of some underlying technology such as Microsoft's COM to be the defining criterion for component. Quite apart from these conceptual categories is the question of size. Some consider components to be the small-scale equivalent of objects in object-oriented programs, while others consider components to be the large-grained equivalent of subsystems or larger (Bass et al. 2000).

To clarify this confusion, in this study several definitions of software components provided in the literature are integrated; these definitions emphasise different aspects of components, such as interfaces, the component market, replacement and reusability (discussed below).

**The concept of components**

The main concepts behind components are shown in Figure 11.

**Figure 11: The main concepts behind components (adopted from Alexandersen et al. 2003)**



- **Components** are units of independent production, acquisition, and deployment that interact to form a functioning system (Szyperski 1998).
- **Components** are executable units of code that provide a set of services through a specific interface (Vitharana 2003). They offer a precisely defined set of services to their environment. The combination of these services is referred to as the component interface.
- **Components** are self-contained units. Data and process are encapsulated in a component - each component stores its own data (as opposed to a database coupling in traditional structured software development).
- **Components** are replaceable parts of a system (Crnkovic and Larsson 2002)

Being self-contained and replaceable units that communicate and connect with other components via interfaces, components could be reused in a number of products, and be replaced by more recent and advanced versions of components in a 'plug-and-play' manner, as long as the interfaces are across the components comprising the products.

### ***Types of components***

The concept of components within the IS world originated within the area of *technical system components*, but is also applicable to *business (software) components*:

- A *technical component* implements a general function/service, independent of a business domain. For example, a technical component could be a network socket component that offers the service of transmitting a file to another point in a network.
- A *business component* implements an autonomous business concept / function or business process. For example, a business component could be a business function such as Human Resources, or an Accounting function in an ERP system that supports a variety of business processes for the corresponding

organizational unit. Or it could be a mathematical function component that provides a specific mathematical function.

Components can be bought, installed and integrated locally, or they can be hosted by a third party. For example, a currency converter is a business component typically hosted by a third party and accessed over the Web.

### ***Granularity***

Components can be *fine-grained* to *large-grained*. Typically, technical components are of finer granularity than business components (in system developed using component technology, business components contain fine-grained technical components) (Crnkovic and Larsson 2002; Alexandersen et al. 2003). The possibility to reuse components is considered to be one of the main advantages of CBD. Initially the focus was on reusing fine-grained components. However, the drawback was that search and integration costs easily outweighed the benefits of reuse. Recently the focus has shifted to reusing large-grained components (Alexandersen et al. 2003). The empirical part of the research presented in this thesis covers the development and reuse of components of different levels of granularity: fine-grained components (in the LeCroy and SAP cases, Chapters 5 and 6 respectively) and large-grained components (in the TCS case, Chapter 7).

### ***Communication via interfaces***

Components communicate with each other by sending and receiving messages. A middleware technology is used to route and deliver messages between components. Currently much work is ongoing to standardize message syntax and semantics (Alexandersen et al. 2003). Although some technical standards have emerged, for most business domains semantic standards are still at an early stage (Bass et al. 2000; Kim 2002). As a result, component integration usually needs to be done manually. As more domain standards emerge, component integration will

become smoother and plug-and-play components may become more of a reality (Alexandersen et al. 2003).

Usually software components need to interact with legacy systems that generally do not have interfaces and clear service specifications. In these cases, a component wrapper has to be built, which exposes the functionality of the legacy system so that it can be viewed as a component (Kim 2002; van den Heuvel et al. 2002).

### ***Deployment***

A software component can be deployed independently and is subject to further assembly by third parties (Pfister 1997). Therefore, for proper integration and functioning, extensive information about components is needed. Component documentation should clearly specify interfaces (the services they provide), (encapsulated) functionality, and the states of components and in which state and which function they could be used. Furthermore, non-functional properties, such as the reliability, performance, security (and pricing, if intended for sale) should also be specified in the component documentation. Although components should be designed to be as independent as possible, they often require the services of other components to function. Thus, context requirements should be explicitly stated in the component documentation. Sometimes, examples of use and reference models (e.g. if used for calculations) that the application is based on, should be documented as well.

#### **2.4.3.2 CHARACTERISTICS OF A COMPONENT-BASED SYSTEM**

Typically, software systems have a long lifetime of at least several years, during which upgraded software systems with more features are released under different release versions. Changes, improvements, and enhancements leading to new software design releases cause a software system to evolve (Peters and Pedrycz 2000). As a result, a software system is updated and changed many times over the period of time that a system lives (Crnkovic and Larsson 2002). Therefore, the

main aim in a software design process is to provide a clear and relatively simple structure of a system, which is *flexible* (facilitating changes to accommodate new needs), *extensible* (essentially open and easily revised to satisfy increasing demands or additional services), *portable* (can be made to execute on different platforms with reasonable effort), and *reusable* (the architecture can be extracted from one application and inserted into a new application with reasonable effort) (Peters and Pedrycz 2000).

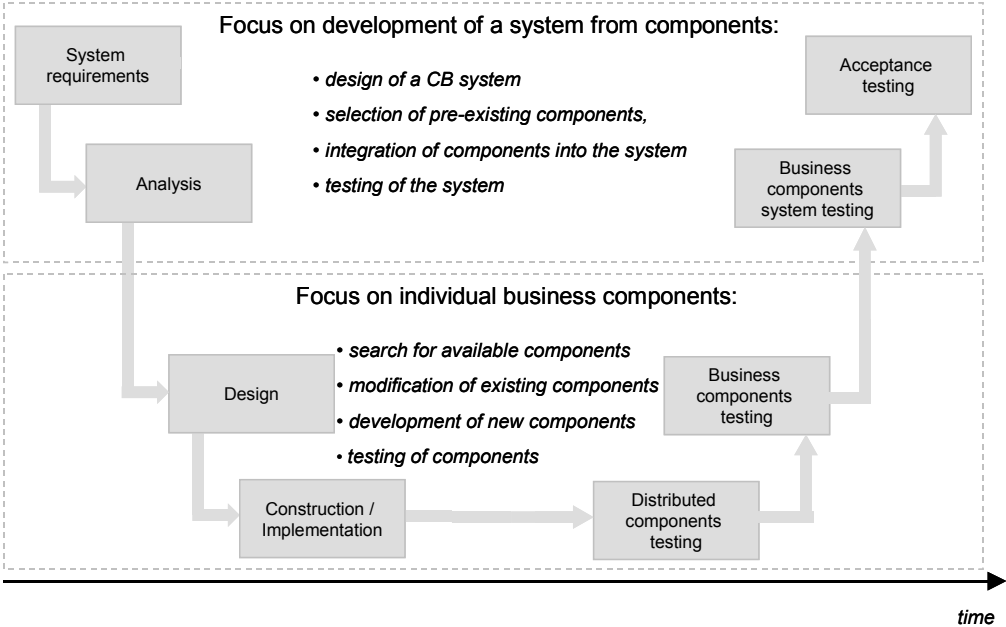
In terms of system structure, CB system architecture is considered to be key to the success of systems with a long life cycle (Crnkovic and Larsson 2002). As compared to a monolithic software system, a CB system is considered to be more flexible, extensible, and reusable (Crnkovic and Larsson 2002). Furthermore, a CB system is easier and more effective to maintain, because it can be maintained in parts (by components), as opposed to a monolithic system which needs to be maintained as a whole (Verbraeck et al. 2002).

#### **2.4.4 THE CBD APPROACH: PROCESS OVERVIEW**

The Waterfall model is too rigid and linear, therefore it does not support iterations, parallel development, incremental delivery or flexibility in software development supported by component technology (Graham et al. 1997). An approach that is common for CBD is referred to as *V-cycle*: it defines the main steps in CBD, which are requirements, analysis, design and implementation, and four different levels of testing – for each of the four previous steps (Herzum and Sims 2000). Figure 12 illustrates the main steps of CBD as a V-cycle approach. The V-cycle is a simplified and linearized representation of a complex development process reality; however it has proved a convenient way to introduce many concepts and deliverables of the CBD process (Herzum and Sims 2000).



**Figure 12: V-cycle approach: CBD lifecycle (modified from Herzum and Sims, 2000)**



As Figure 12 illustrates, the first two and last two stages, i.e. system requirements, analysis and two corresponding types of testing, focus on a *system*. These stages are concerned with the design, assembly, and testing of a system using business components. It can also include selection of pre-existing components that can be reused in the system. By contrast, the intermediary four stages, i.e. design, implementation and two corresponding types of testing, focus on *business components*<sup>6</sup>. These stages concerned with the designing, building and testing of

<sup>6</sup> Herzum and Sims (2000:556) define *business component* as ‘the software implementation of an autonomous business concept or business process. It consists of all software artefacts necessary to represent, implement, and deploy a given business concept as autonomous, reusable element of a larger distributed information system’. Business components are large-grained. They contain fine-grained components (referred to by Herzum and Sims (2000) as ‘*distributed components*’, defined as ‘a software artefact that can be called at run-time with clear interface, and clear separation between interface and implementation’).

individual business components (Herzum and Sims 2000). Design and implementation can also include the search for available components, internally or from component markets, and the customization of components to suit the needs of a system being developed.

The next section will elaborate on the benefits and challenges associated with CBD.

#### **2.4.5 REUSE IN CBD: BENEFITS AND CHALLENGES**

Initially, the CBD methodology has been presented as a revolutionary approach to software development, promising dramatic improvements in software development efficiency (Huang et al. 2003; Vitharana 2003). The main advantage expected from adopting CBD methodology is the possibility to reuse components (Bass et al. 2000; Kunda and Brooks 2000; Crnkovic and Larsson 2002; Ravichandran and Rothenberger 2003; van Hillegersberg 2003; Vitharana 2003).

A company developing a CB system can apply different modes of component reuse in terms of *source* of components:

- Internal reuse - reuse of components developed in-house (e.g. Crnkovic and Larsson, 2002);
- Reuse from component markets - reuse of components procured from component markets to develop the CB system (Traas and van Hillegersberg 2000);

Hybrid reuse - combination of internal reuse and reuse from component markets (e.g. Homann et al. 2004).

Furthermore, the reuse concept can be used on different levels in terms of *granularity* (Herzum and Sims 2000; Crnkovic and Larsson 2002):

- reuse of small-size components (high granularity, mostly technical components);
- reuse of business components (encapsulating business functions);

- reuse of complete products that are integrated in complex systems.

Crnkovic and Larsson (2002) argue that, on each level of reuse, the demands on the reusable components, on the component management and on the integration process are different.

The possibility to reuse components influences directly the efficiency of the development process, in particular:

*Better quality* - reliability of products increases when components are reused in a number of products (Crnkovic and Larsson 2002; van Hillegersberg 2003; Vitharana 2003).

*Improved productivity* - shorter time-to-market, because, first, a new product could be developed faster when (some) components are reused in a number of products. Second, the variety of products increases because it is easier to customize, upgrade, and add new features to existing products (Bass et al. 2000; Kunda and Brooks 2000; Crnkovic and Larsson 2002; Kim 2002; Huang et al. 2003; Ravichandran and Rothenberger 2003; Vitharana 2003). Furthermore, development time could be reduced because components could be developed in parallel independently by teams located in the same building or at remote locations (Repenning et al. 2001).

*Lower costs* – development costs are lower when components are reused in a number of products (Kunda and Brooks 2000; Ravichandran and Rothenberger 2003; van Hillegersberg 2003). Moreover, component markets provide an alternative to in-house development: the components are procured from component markets for a lower price than they would cost to develop in-house.

However, empirical research on CBD has challenged these benefits and shown that ‘it often took longer to develop a reusable component than to develop a system for a one-off purpose’ (Huang et al. 2003). It was argued that the benefits

are difficult to achieve in the first place, and that they cannot be achieved immediately, but in the long run (Crnkovic and Larsson 2002). Empirical research on co-located CBD reported a number of challenges that companies faced, trying to achieve the benefits of reuse:

- Before the components can be reused, a sufficiently large pool of reusable components needs to be developed (Bass et al. 2000).
- Often, there is a gap between requirements and available components. If some components can be found neither in-house nor on the component market, then cost-benefit analysis, and possibly negotiation with a customer, is needed to decide whether to adjust the requirements to the available components or to develop customised components (Vitharana 2003).
- There are a multitude of component repositories (Traas and van Hillegersberg 2000). Therefore, effective classification and coding schemes are needed in order to develop advanced searching mechanisms and enable component seekers to locate components (Vitharana 2003).
- Often, requirements of the components are not well understood, which brings an additional level of complexity. Crnkovic and Larsson (2002) explain that long-life products are most often affected by evolution of different kinds: evolution of system requirements; evolution of technology used in software products and other related domains; business changes and organizational changes (Grinter 1998). ‘As a result of new requirements for the products, new requirements for the components will be defined. The more reusable a component is, the more demands are placed on it’ (Crnkovic and Larsson 2002).
- Stable standards for component technology and certified components are lacking (Bass et al. 2000; Kim 2002) and components from different producers are often incompatible.

- A need to decide on the level of granularity of components (large- or fine-grained), to achieve (i) a higher reuse rate (for internal reuse in-house), and (ii) a higher demand (for commercial components) (Alexandersen et al. 2003; Vitharana 2003).
- A need to decide how generic (or how specific) a component should be (Vitharana 2003). On the one hand, to be reused, a component has to be generic enough to be appropriate for different products. On the other hand, if a component is too generic, it might not be reused at all if it is not associated with any particular business or functional domain.
- A need to decide on required documentation: what should be included in the documentation?
  - Interfaces – how should interfaces be described? Should UML be used for interface documentation (van Hillegersberg 2003)? There are no widely accepted standards and guidelines about this (Bass et al. 2000; Vitharana 2003).
  - How detailed should documentation be? Documenting in-house developed components for internal reuse takes time and is often considered as an administrative overhead. However, documentation is needed to ensure that a component and the logic behind it can be understood in case modifications or bug fixes are needed.
  - Should documentation include the source-code (i.e. ‘white-box’ documentation) (van Hillegersberg 2003)? On the one hand, the source-code could be used for understanding how a component functions and could increase the sales of the component because customers can see what they are buying. On the other hand, revealing the source-code is a potential threat to the intellectual property of an organisation.

- A need to decide on building vs. buying: to buy a component from the component market or build a needed component in-house (Vitharana 2003)? A cost-benefit analysis is required to evaluate and compare the alternatives. Developing components in-house might take longer and cost more. However, searching for required components to buy could also take time and cost money; components available from the market might require some modification, and their price might be per use (i.e. per product, so each internal reuse costs money).

As described above, the empirical research on CBD reports on the difficulties in achieving reuse that challenge the potential benefits. This literature is mostly based on case studies in co-located CBD projects. Table 3 gives an overview of the core literature on the management of CBD discussed in this chapter (marked as ‘V’): (i) context (co-located or globally distributed); (ii) research approach (case study or industry survey); (iii) assessment of benefits and challenges in CBD; and (iv) aspect of CBD methodology discussed.

**Table 3: Overview of core literature on management of CBD**

	(Bass et al. 2000)	(Kunda and Brooks 2000)	(Crmkovic and Larsson 2002)	(Ravichandran and Rothenberger 2003)	(van Hillegersberg 2003)	(Vitharana 2003)	(Traas and van Hillegersberg 2000)	(Kim 2002)	(Carmel 1999:27-32)	(Huang et al. 2003)	(Alexandersen et al. 2003)	(Turnlund 2004)
Context of CBD:												
<b>Globally Distributed CBD</b>					V		V		V		V	V
Co-located CBD	V	V	V	V		V		V		V		
Research approach:												
Case study		V	V		V			V	V	V	V	V
Industry survey	V						V					
Theoretical framework				V		V						
Benefit (+), challenge (-), both (±):												
Reuse	±	+	±	±	-	-	+			-	+	
Better quality			+		+							
Shorter time-to-market	+	+	±	+	+	+	+	+		±		-
Lower development costs	+			+	+		+					
Management of CBD (focus):												
Tools and methods	V		V		V			V				V
Social issues		V								V		
Inter-site coordination						V			V			V
Component markets				V			V					
Strategic aspects											V	

In the light of globally distributed software development, it has been argued that CBD enables each site to take ownership of particular components and work on them independently without much need for inter-site communication and coordination (Carmel 1999; Colbert et al. 2001; Reppenning et al. 2001). Thus, the adoption of CBD by organisations involved in globally distributed projects might ease coordination problems faced in traditional (non CB) GSD projects (discussed in Section 2.2.1) caused by geographical dispersion, time-zone and cultural differences. On the other hand, the difficulties of achieving reuse reported in co-located CBD projects might be relevant in GD CBD projects as well.

The next section will discuss how to organise and manage GD CBD successfully, based on an overview of the existing literature on the management of CBD in a co-located and globally distributed environment.

#### **2.4.6 HOW TO ORGANISE AND MANAGE CBD IN A GLOBALLY DISTRIBUTED ENVIRONMENT**

Despite the fact that increasing numbers of companies are setting up software development in a globally distributed environment and at the same time are adopting a CBD methodology, research on the management of GD CBD is just emerging. As Table 3 (in Section 2.4.5) shows, the majority of research on the management of CBD is conducted on co-located settings, and at present little is known about the management of GD CBD.

This section will give an overview of issues addressed in the literature on the management of CBD, co-located and globally distributed; and will discuss potential factors contributing to success in GD CBD. These factors cover (I) *Inter-site coordination in GD CBD*; (II) *Tools and methods to support CBD*; (III) *Human, social and organizational issues in CBD*; and (IV) *Knowledge sharing in CBD*. To distinguish between the findings from co-located and from globally



distributed CBD, each factor/issue is discussed first as addressed in the co-located CBD, followed by the findings from GD CBD research.

## **I Inter-site Coordination in GD CBD**

### ***Co-located***

As discussed earlier, research on co-located CBD suggests that components can be developed remotely with minimum coordination across dispersed locations (Carmel 1999; Colbert et al. 2001; Repenning et al. 2001).

### ***Globally distributed***

Despite the expectation that components could be developed without much need for inter-site communication and coordination, existing studies of GD CBD point out that in GD CBD still much coordination between sites is required (Carmel 1999; Turnlund 2004). For example, Carmel (1999) describes the difficulties faced by IBM in a globally distributed project developing software based on JavaBeans component technology. Initially, IBM tried to organise ‘follow-the-sun’ development, so that during the USA daytime the USA headquarters site set up specifications for each JavaBean and assigned it to one of the remote locations (in China, Belarus, Latvia or India). Then, the remote locations worked on the code during their daytime and by the end of the day (by the morning in the USA) sent it back to the USA site for successive rounds of reviews and feedback. After testing in the USA, instructions were sent to the remote location for the next iteration. However, this arrangement did not work because the USA site was handling too many tasks sending components back and forth. As a result, instead of a ‘follow-the-sun’ approach, the ownership of components was delegated to the remote sites, and the USA headquarters role was reduced to managing the complicated coordination process. Carmel (1999:32) suggested that ‘the essence of making this complicated coordination process work was a good collaborative technology infrastructure’.

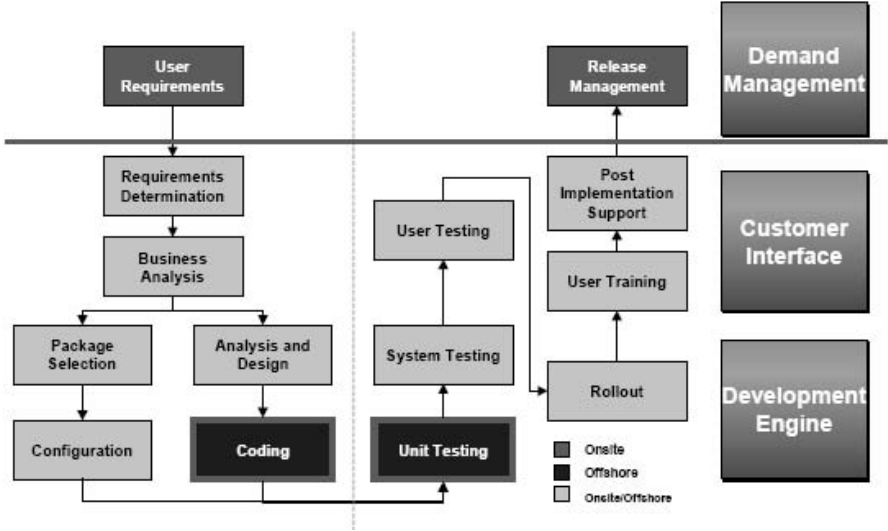
Furthermore, it is reported that in a globally distributed environment, granularity and interdependencies between components become an issue. For example, Turnlund (2004:30) pointed out that:

At the surface level it is attractive to push every part of the system down into its own granular, self-contained entity. With a single physical location for development, a group can execute within this model. From a combinatorial aspect (geography, number of interconnects, variability of execution) this ‘trust everyone to understand the overall system’ method becomes a disaster.

When too many relatively complex interrelationships need to be managed, effective parallel development is not possible any more: in this case the ‘integration exercise becomes a complex, rework-ridden, lengthy, indeterminate majority of the development exercise’ (Turnlund 2004:29). To reduce the complexity involved in dealing with too many fine-grained components across remote sites, Turnlund (2004) suggests ‘logical geographic groupings of component control’, so that each site can take care of its own components. This attempt to divide work between sites is similar to the approach to division of work in traditional GDSD, aiming to reduce interdependencies between work packages assigned to different sites (see ‘Division of work’ in Section 2.2.2.2).

However, dispersed teams can do only limited types of work independently: the majority of tasks still require a great deal of communications and coordination between the teams. For example, Figure 13 illustrates the division of work between *onsite* (customer site of Skandia Financial Concepts (SFC) in Zurich, Switzerland) and *offshore* (development centre of TATA Consultancy Services (TCS) in Gurgaon, India) for the development of Skandia’s financial services platform, described by Alexandersen et al. (2003).

**Figure 13: Division of onsite/offshore work for development of the Skandia platform (adopted from Alexandersen et al. 2003)**



Alexandersen et al. (2003:17) explained the criteria that the division of work between onsite, onsite-offshore, and offshore work were based upon:

First there was the need for direct customer contact and at the location of the customers. Thus user requirements and release management were primarily *onsite* activities. Second, those activities that were self contained, and could be conducted with minimal customer contact, such as coding and unit testing, were sent *offshore* to take advantage of the cost structures, quality, and availability of offshore personnel. Finally, activities that required co-work by client, SFC, and technical TCS personnel were conducted in a *mixed onsite-offshore* mode with frequent mediating contacts through communication technologies and site-visits.

As can be seen from Figure 13, the majority of activities required extensive communications, and onsite and offshore teams conducted them jointly. Only coding and unit testing was done independently at the offshore location.

## II Tools and Methods to support CBD

The majority of the literature on co-located CBD talks about the importance of automated tool support for successful CBD. Tools required for CBD include (i) tools for the development and management of components, (ii) configuration and version management tools, (iii) tools for tracing and tracking bugs, and (iv) tools for testing. In order to support CBD these tools need to provide the following capabilities:

---

### 1. Development environment

---

#### *Co-located*

Research on the co-located CBD reports that development environment support is essential to enable editing, compiling, building, debugging and testing of components. Although recently more tools for CBD have emerged, the early adopters of CBD had to develop their own tools and integrated development environments to manage large scale CBD. For example, the company described by Crnkovic and Larsson (2002) developed their own software development environment: ‘an internally built program package which encapsulates different tools, and provides support for parallel development’.

#### *Globally distributed*

The literature on management of GD CBD does not mention the need for a development environment, but it is likely that in GD CBD it might be more difficult to provide development environment support than in co-located CBD because the development environment would need be connect to remote sites. This leads to the question: *how can development environment support be provided in GD CBD?*

In GD CBD, dispersed sites can be connected either under a single development environment, or by synchronizing files across locations in a Web-based environment in a peer-to-peer distributed system. As opposed to the Open Source

community which uses Web-based IDEs such as SourceForge to work in a distributed environment, commercially available IDEs (e.g. Java IDE) and modelling tools (e.g. Rational Rose) are usually client-server based systems designed primarily for a single user (i.e. only one person can work on a chunk of code or software design at a time). They offer only limited support for distributed development (Herrera 2002). Therefore, companies and researchers are working on how to design collaborative capability into IDEs (e.g. Cheng et al. (2004), and the research of John Grundy on JViews multiple user system, a component-based software architecture for multiple view)<sup>7</sup>.

---

## **2. Automated management of interdependencies between components**

---

### ***Co-located***

In CBD there is a need to manage interdependencies between components: ‘when it is about complex products, it is impossible to manually track dependencies between the components’ (Crnkovic and Larsson 2002), and an automated tool for checking consistency is needed.

### ***Globally distributed***

This issue is not addressed in the literature on the management of GD CBD. It is likely that in a globally distributed environment the need for automated tools to manage inter-dependencies between components becomes even more critical for success, because, in addition to managing interdependencies, components need to be synchronised across sites. This leads to the following question: *how can components and their interdependencies be managed in a globally distributed environment?*

---

<sup>7</sup> From the home-page of Prof. John Gundy

<http://www.cs.auckland.ac.nz/people/profile.php?id=jgru001>

---

### **3. Version and configuration management (tracing and tracking of components)**

---

#### ***Co-located***

It is suggested that the development of reusable components requires support for the development and maintenance of different versions of components for different product versions (Crnkovic and Larsson 2002)<sup>8</sup>. Therefore, version and configuration management process support is needed on two levels. First, on the component level, component versioning needs to facilitate the tracing of each component from inception to delivery, and (versions of) components used in multiple applications need to be coordinated. Second, on the product-integration (application) level, different versions of a product would have a different set of components, and different versions of components which need to be managed consistently (Crnkovic and Larsson 2002; Vitharana 2003). Furthermore, third party components (e.g. from the component market) used in a product have their own versioning that needs to be managed as well.

This requires an advanced version and configuration management process to be defined, and powerful Software Configuration Management (SCM) tools<sup>9</sup> to support this process (Crnkovic and Larsson 2002; Vitharana 2003).

#### ***Globally distributed***

Existing SCM tools are designed for distributed development. The literature on co-located CBD suggests that SCM tools are needed to support people working in

---

<sup>8</sup> It is important to note that version and configuration management is different from the automated management of interdependencies between components, discussed earlier. The former is concerned with tracing and tracking of versions of components, while the latter is concerned with technical interdependencies between components such as interfaces and messages (service request and service response).

<sup>9</sup> Version Control System (VCS) is part of SCM tools

parallel from different work stations located at the same (or different) buildings (Crnkovic and Larsson 2002). However, in a globally distributed environment a more powerful network and server(s) are needed, and differences in tools or different versions of the same tools used at remote sites might cause difficulties such as lack of compatibility between files/versions developed at different sites. This lead to the question: *how can version and configuration management be supported in GD CBD?*

---

#### **4. Bug tracing and tracking**

---

##### ***Co-located***

It is suggested that in co-located CBD there is a need to trace bugs on three different levels: first, on the *system* level, where customers report problems with their specific product; second, on the *product* level, where errors are detected in a specific product version; and finally, on the *component* level, where the fault is located (Crnkovic and Larsson 2002). The need to trace bugs is closely related to version and configuration management, because ‘a modification of the component can lead to an explosion of new versions of different products which already exist in several versions. The relations between components, products and systems must be carefully registered to make possible the tracing of errors on all levels’ (Crnkovic and Larsson 2002). This introduces additional requirements for an advanced SCM tool. Crnkovic and Larsson (2002) have indicated that the need to manage and maintain complex products is not unique to CBD: however, ‘what is specific to the component-based approach is the mapping between products and components and the management of error reports on product and component level, the most difficult part of the management’ (Crnkovic and Larsson 2002).

##### ***Globally distributed***

This issue is not addressed in the research on management of GD CBD. However, in practice, there are several tools that have Web-based interfaces, such as

Rational ClearQuest. This, for example, allows customers to report problems using Web-based forms, independent from their geographical location, and to track the status of the problem (i.e. progress in resolving the bug reported).

In addition to a Web-based system to report bugs, companies involved in GD CBD need to be able to trace back bugs on product and component levels (as explained above) across globally dispersed locations. This leads to the question: *how can bug tracking and tracing be supported in GD CBD?*

---

## 5. Testing and quality assurance

---

### *Co-located*

Research on co-located CBD suggests that comprehensive tools and techniques are required for different types of testing:

- component (unit) testing: ‘almost like an individual application, though on a smaller scale, each component must undergo verification and validation testing throughout its development process’ (Vitharana 2003);
- application (integration) testing; and
- system testing.

Furthermore, in order to assure the quality of a final product, quality certification is needed for third party components used in the product (Vitharana 2003).

### *Globally distributed*

This issue is not mentioned in the literature on the management of GD CBD. Typically, tools for testing would be part of the development environment. Testing in a globally distributed environment might be more difficult to organise than in a co-located one, because components developed at remote sites need to be tested together (in particular for application and system testing). This leads to the question: *how can testing be supported in GD CBD?*



In addition to the tool capabilities discussed above, the literature on co-located CBD mentions the need for a commonly accepted standard method for CBD.

---

## **6. Methods for CBD**

---

### ***Co-located***

The literature on co-located CBD suggests that ‘what is needed by the CBD project team is a commercial-level CBD methodology that covers a whole lifecycle process and provides practical guidelines’ (Kim 2002).

There are several CBD methodologies, such as Catalysis (D'Souza and Wills 1999) and Componentware Methodolog<sup>10</sup>. Furthermore, Firesmith and Henderson-Sellers (2001) have proposed OPEN Process Framework - a meta-process that can be tailored to a CBD approach. Existing methodologies are based on different technical standards and different component technologies, while a commonly accepted standard reference model for an engineering method to consistently guide CBD is lacking (Bass et al. 2000; Kim 2002).

At present companies are trying to find their own way to succeed in CBD. For example, in order to support CBD and facilitate component reuse in Korea, a nationwide Component Industry Promotion (CIP) project was launched by the Korean Ministry of Information and Communication (Kim 2002). Kim (2002) reported that the CIP project developed a standard reference model for ‘a whole lifecycle CBD methodology’ that comprised four main stages of CBD: (1) planning the project and comprehending requirements, (2) developing components, (3) developing an application using existing components, and (4) deploying a CB application. Each stage was broken into phases and further into activities. The reference model combined the main features of existing methods,

---

<sup>10</sup> Available on Carnegie Mellon Software Engineering Institute web site:  
<http://www.sei.cmu.edu>

such as Catalysis and Componentware, and added new techniques based on CBD projects in Korea which participated in the nationwide CIP project.

Taking into account the difficulties related to component reuse (discussed in section 2.4.5), Crnkovic and Larsson (2002) suggest:

The reuse orientation requires a systematic approach in design planning, extensive development, support of a more complex maintenance process, and in general more consideration being given to components. It is not certain that an otherwise successful development organisation can succeed in the development of reusable components or products based on reusable components.

A standard CBD method / methodology would provide a structured and systematic approach for CBD and would facilitate reuse.

### ***Globally distributed***

Methods for GD CBD are not addressed in the literature. This leads to the question: *What methods can support the lifecycle of GD CBD, in particular aspects that are unique to GD CBD, such as methods to support division of work between site, integration procedures?* For example, to support working in a globally distributed environment, these methods should be web-enabled.

## **III Human, social and organizational issues in CBD**

### ***Co-located***

It is suggested that designing components requires unique skills that involve in-depth knowledge of CB technologies, design principles and decisions different from those used in traditional software development (Vitharana 2003). Furthermore, CBD allows separation of skills which in turn results in new roles, for example infrastructure builder, component developer or application assembler (Bass et al. 2000). Therefore, in companies/teams who switch from traditional software development to CBD, top management needs to invest significant

resources in retraining current personnel and/or hiring new personnel (Vitharana 2003).

Kunda and Brooks (2000) have studied the adoption of CBD from a socio-technical perspective, aiming to identify (i) problems experienced by organizations implementing CBD on individual, group and organization levels, and (ii) factors influencing CBD success. Based on case studies in three companies adopting CBD, Kunda and Brooks (2000) report problems related to cognitive skills, disincentives, organizational politics and organizational culture. They identified factors that affect CBD success as follows:

- *Human factors*, which include motivation, enthusiasm, incentives, cognitive skills, and customer ownership.
- *Social factors* are: different perceptions, different goals, and interactions and communication between group members.
- *Organizational factors* are: political issues, organizational and business strategy; organizational resources and support; organizational setting and management style, and organizational culture. These results are supported by Huang et al. (2003), who studied the importance of organizational cultures and sub-cultures in the success of CBD adoption.

Despite the findings of Kunda and Brooks (2000) regarding the importance of human, social and organizational issues in the success of CBD, the majority of the literature on the management of CBD emphasizes the importance of tools and technologies for successful CBD, while the social and human issues involved in CBD are typically neglected (Kunda and Brooks 2000; Huang et al. 2003).

### ***Globally distributed***

Research on the management of GD CBD does not address issues related to human, social and organizational factors. It is possible that the same problems related to cognitive skills, disincentives, organizational politics and organizational

culture identified in co-located CBD projects will be faced by globally distributed teams. Moreover, the implications of global distribution, such as geographical distance and cultural differences, may make these problems more severe. This leads to the following questions: *what is the impact of human, social and organisational factors on success in GD CBD*, and *how should these factors be managed in GD CBD?*

#### **IV Knowledge sharing in CBD**

##### ***Co-located***

Huang et al. (2003) report that intensive knowledge sharing and collaboration throughout a whole development life cycle are required for the successful adoption of CBD. The authors suggest that sharing knowledge and creating ‘knowledge redundancy’ is ‘a critical step in reducing conflict resulting from misunderstandings between and within stakeholder groups’ (Huang et al. 2003:96) involved in the development process.

##### ***Globally distributed***

This issue is not addressed in the research on management of GD CBD. Taking into account that misunderstandings and conflicts often happen in globally distributed teams, it is likely that in GD CBD knowledge sharing is more difficult to achieve in GD CBD. This leads to the questions: *what is the impact of knowledge sharing on success in GD CBD*, and *how can knowledge sharing be facilitated GD CBD?*

This section has given an overview of what is known about the management of CBD, co-located and globally distributed, based on the existing literature. Taking into account that only a limited number of studies on GD CBD have been published, the findings from research conducted in co-located CBD were applied

to globally distributed projects, suggesting possible scenarios and posing questions that need to be addressed in the context of GD CBD.

The next section will describe different measures used to evaluate the success of IS projects. Then, potential factors that may contribute to success in GD CBD will be discussed in Section 2.6.

## 2.5 SUCCESS IN INFORMATION SYSTEM PROJECTS

Success in IS development projects has been studied from various angles. Some studies put the emphasis on the project outcome to assess success: for example, product delivery being on time and within the budget (Nelson and Coopriider 1996; Hoegl and Gemuenden 2001; Nellore and Balachandra 2001), and product and process quality. Others focus on the quality of interactions between project members to assess collaboration success, such as communications and team performance (Nelson and Coopriider 1996; Hoegl and Gemuenden 2001).

In this sense, success is represented in this research as a combination of product outcome, people-related outcome and collaboration process quality.

Product success can be represented by various indicators, such as growth in sales, product delivery on time and within the budget (Nellore and Balachandra 2001; Andres 2002), short time-to-market (Datar et al. 1997) and increase in reuse of components (Crnkovic and Larsson 2002). In line with these indicators, *product success* is thus defined as the achievement of project objectives (Gallivan 2001).

This criterion for product success can either be objective, i.e. based on market or company data, or subjective, i.e. based on project participants' perceptions of product success.

While the IS literature on globally distributed teams has traditionally focused on technical tools, such as ICT, and their contribution to success (Carmel 1999; Karolak 1999; Herbsleb et al. 2002), some hints about other factors affecting

product or project success have been provided by past studies that mainly focused on co-located teams. Among these factors, research has suggested knowledge sharing (Nelson and Coopriider 1996), informal communications and personal relationships (Hoegl and Gemuenden 2001), interactions between parties involved in the development, for example, customers or marketing and engineering specialists (Nelson and Coopriider 1996), and team cohesion (Rafii 1995; Hoegl and Gemuenden 2001).

A desired result of a distributed team can also be a people-related outcome (Hoegl and Gemuenden, 2001) which entails meeting the psychological needs of the members (Gallivan, 2001). Hoegl and Gemuenden (2001) and Gallivan (2001), for example, suggest that, in addition to performance objectives, teams must also work in a way that increases members' motivation to engage in future teamwork. There should be some level of personal satisfaction that motivates individuals and teams to continue their engagement in collaborative work despite geographical, time and cultural differences.

In this research *personal satisfaction* is perceived as the outcome of a positive social experience. Such positive social experience can, for example, be in the form of stress-free communication rituals between remote counterparts and collegial relationships between remote teams.

Some factors that may foster people-related outcomes and thus may improve personal satisfaction are open and multiple informal communication channels (Hoegl and Gemuenden, 2001), the encouragement of interactions between parties involved in the development process (Nelson and Coopriider, 1996), and the cohesion of a team (Gallivan 2001; Hoegl and Gemuenden, 2001).

The success of a distributed team can also be assessed in terms of the quality (efficiency) of a process through which dispersed team members collaborate.

The word ‘collaboration’ comes from the Latin words *com* (prefix *together*) and *laborare* (verb *to work*). It means that two or more individuals work jointly on an intellectual endeavour (Webster 1992).

**Successful collaboration** is a complex, multi-dimensional process characterized by constructs such as coordination (Faraj and Sproull 2000), communication (Weick and Roberts 1993), and structure (Scott 1992; Adler and Borys 1996), which achieves a predefined goal (a product or desired performance) through group effort.

Furthermore, OB studies stress the importance of social aspects such as relationships (Gabarro 1990), trust (Meyerson et al. 1996) and shared meaning (Donnellon et al. 1986; Bechky 2003) for successful collaboration.

Naturally, geographical, cultural and time-zone differences pose additional challenges to globally distributed teams to achieve success, whether seen as a people-related outcome, a product outcome or a collaboration process quality. Managerial practices that involve inter-site coordination mechanisms and technologies (discussed in Section 2.2.2) help to reduce geographical, cultural and time-zone differences, and problems and breakdowns associated with these differences / gaps (discussed in Section 2.2.1). Therefore, in addition to product, process and people-related measures of success commonly used in IS research, in this thesis the success of GD CBSD is assessed based on the degree of success in bridging gaps between globally distributed teams.

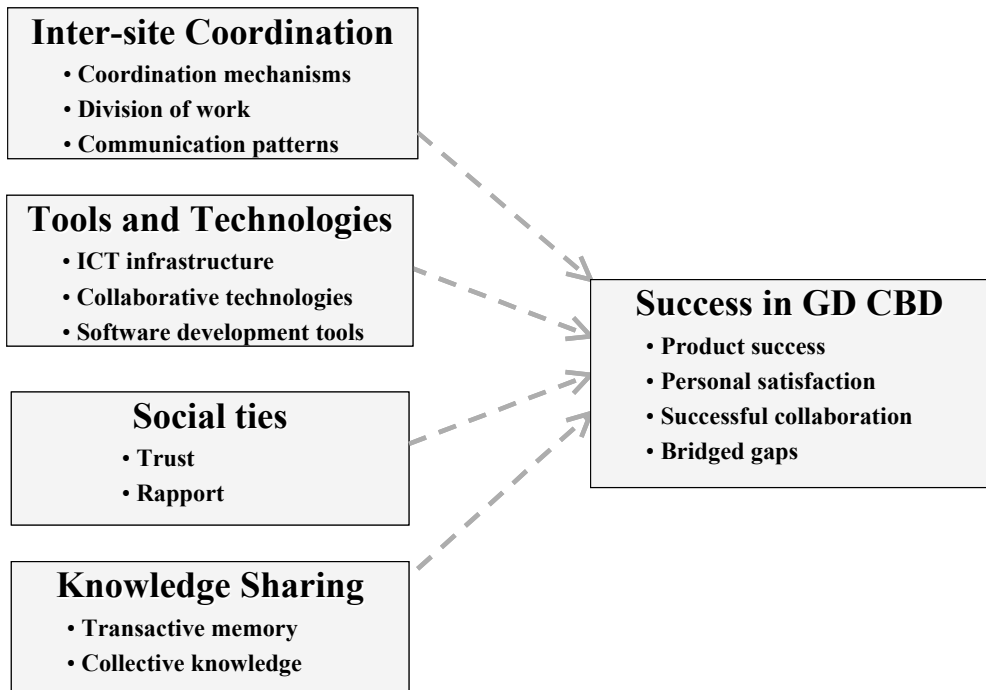
In this research **bridging of gaps** between globally distributed teams is assessed based on the perceptions of team members of gaps (geographical, time-zone and cultural) as being a problem.

## 2.6 CONCLUSION: POTENTIAL SUCCESS FACTORS

Past research in the IS field stresses the importance of tools, technologies and coordination mechanisms for successful GSD projects (Carmel 1999; Karolak 1999; Herbsleb et al. 2002). Despite the fact that research on the management of GD CBD is very limited, existing studies on GD CBD recognise the importance of coordination mechanisms and technologies for success in GD CBD. Furthermore, the OB literature offers several factors, such as social ties and knowledge sharing, that may positively affect success through social activities and personal interactions.

Figure 14 illustrates schematically the potential factors that may contribute to success in GD CBD. These factors combine potential success factors identified in the research on traditional GSD (Section 2.2.3), research on GD teams (Section 2.3.3) and research on co-located and globally distributed CBD (Section 2.4.6).

**Figure 14: Potential factors that may contribute to success in GD CBD**





Chapter 2 has given an overview of the existing literature on GD CBD and other related research areas, and identified gaps in the literature that motivated and inspired the research project presented in this thesis. Chapter 3 will present and explain the theoretical lens that accommodates the potential factors contributing to success in GD CBD suggested above (Figure 14), and serves as a basis for empirical investigation.

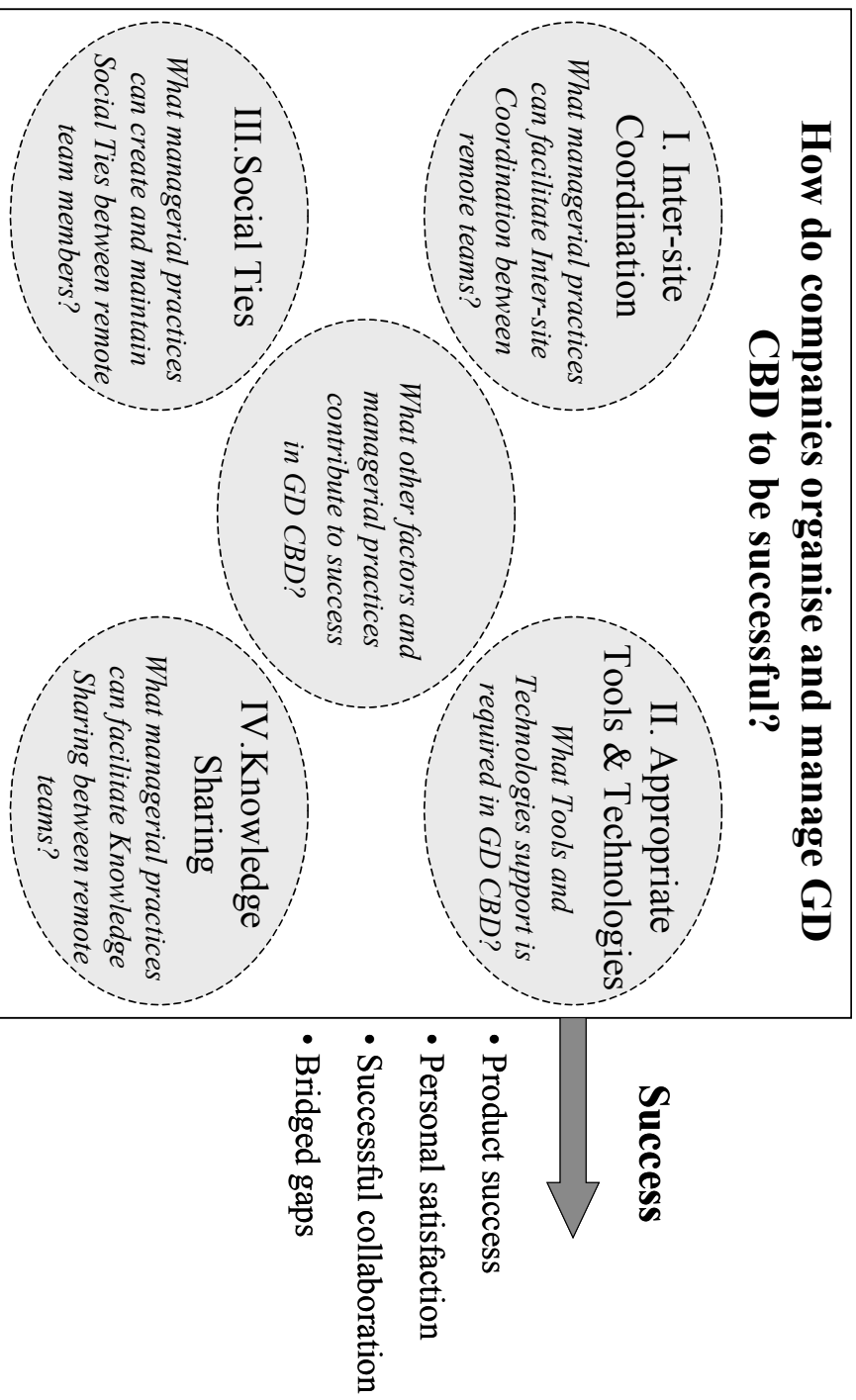
### **CHAPTER 3 RESEARCH FRAMEWORK: THE THEORETICAL LENS FOR EMPIRICAL INVESTIGATION**

The research question *'how do companies organise and manage Component-Based Development in a globally distributed environment to be successful?'* aims to explore in depth the phenomenon of GD CBD.

To address this question an empirical investigation into GD CBD projects is undertaken. It is important to note that the majority of the literature on the management of CBD, both co-located and globally distributed, was published in years 2002 to 2004, while the empirical data collection for this research started in November 2001. At that time, the only available theoretical input to study the phenomenon of GD CBD was from research into traditional GSD and into collaboration in GD teams, based on which the potential factors that may contribute to success in GD CBD were suggested (Figure 14).

Thus, empirical investigation undertaken in this study was exploratory to a great extent: potential success factors suggested in the related research served as a theoretical lens to explore the management practices of GD CBD. The theoretical lens provided a basis for empirical investigation, but did not limit it to only these factors. Figure 15 illustrates the theoretical lens. The theoretical lens suggests that inter-site coordination, appropriate tools and technologies, social ties, and knowledge sharing could be important to, and positively affect, success in GD CBD. The proposed factors are supported by studies from the IS and OB literature outlined in Chapter 2.

Figure 15: Theoretical lens



The empirical investigation was designed to fulfil two objectives:

- First, it aimed to identify what factors contribute to success in GD CBD. Potential factors identified in the theoretical lens (Figure 15) served as a starting point for the empirical research.
- Second, it aimed to collect managerial practices that exemplify how companies organise and manage GD CBD.

This chapter has presented the research framework that was developed to serve as a theoretical lens for the empirical investigation. Chapter 4 will describe the research method applied in this research and explain the research process.



## CHAPTER 4 RESEARCH METHOD AND PROCESS

*Qualitative data are sexy* (Miles and Huberman 1994:1)

This chapter explains the research methodology and describes the research process. Section 4.1 explains the choice of the qualitative case study methodology and describes it. Section 4.2 elaborates on the case selection criteria. Section 4.3 describes the research design and process. Sections 4.4 and 4.5 elaborate on the data collection and analysis respectively. Section 4.6 explains the criteria for assessing the quality of the empirical research.

### 4.1 QUALITATIVE CASE STUDY RESEARCH

*The MIS researcher selects a methodology based on several factors including rigor, relevance, subject area, and personal preferences.*

(Palvia et al. 2003)

According to Yin (1994), *case study research* is appropriate to investigate a phenomenon within its real-life context, to answer *how* and *why* questions, when the investigator has little control over the events. The investigation of GD CBD satisfies these criteria: it aims to address the question ‘*how do companies organise and manage Component-Based Software Development in globally distributed environment to be successful?*’ in the real-life context. The case study method covers both the phenomenon of interest and its context, producing a large number of potentially relevant variables (Yin 1994). Therefore, a case study method was chosen as the most appropriate approach for this research.

The case study method is widely used in IS research. Palvia et al. (2003) examine the usage of different methodologies in MIS research, based on an overview of leading MIS journals during the period 1993-1997. One of the trends they

observed is a greater use of the case study method and other qualitative techniques over the years (Palvia et al. 2003)<sup>11</sup>.

Applying a case study method as a research strategy involves the use of an all-inclusive method and offers several approaches to data collection and analysis (Yin, 1994). Typically, a study based on a case study methodology from an interpretive perspective starts with a discussion of the existing literature followed by numerous data collection methods and a careful analysis of the evidence (Yin 1994), as was done in this research.

Case study research involves gathering evidence from a variety of sources: documentation, archival records, questionnaires, interviews and observations (Eisenhardt 1989; Yin 1994). Triangulation of data collected from multiple sources allows an in-depth study of a phenomenon from different angles and may increase the validity of the research findings.

This research adopted a *qualitative interpretive* approach: empirical evidence collected for this research is qualitative. It is based on words and not numbers (Miles and Huberman 1994). In addition, data analysis methods did not involve any quantitative procedures<sup>12</sup> (Strauss and Corbin 1998).

*Interpretive* research philosophy implies that ‘our knowledge of reality is gained only through social constructions such as language, consciousness, shared meaning, documents, tools and other artefacts’ (Klein and Myers 1999). This philosophy acknowledges different experiences of individuals within the same context and allows in-depth analysis of a unique situation or phenomenon. Research on GD CBD is yet in the very early stages, therefore, interpretive

---

<sup>11</sup> For example, in MISQ, case study is the second most frequently used method (15.7%) after surveys (22.1%) (Palvia et al. 2003).

<sup>12</sup> Strauss and Corbin (1998) define *qualitative research* as any type of research that produces findings not arrived at by statistical procedures or other means of quantification.

research that is closely connected to empirical reality is more appropriate to explore in depth the emerging phenomenon of GD CBD, rather than a positivist approach based on quantifiable measures of variables, hypothesis testing and evidence of formal propositions (Klein and Myers 1999), but loosely connected to empirical reality (Lee 1991). In this research the phenomenon of GD CBD is studied in-depth following the interpretive tradition: then, as one of the outcomes of this research, it suggests propositions about the management practice of GD CBD. These propositions can be tested in a positivist manner in future research.

*Multiple* case studies and subsequent cross-case analysis were conducted to increase the external validity of the research (Yin 1994). Four case studies were conducted in four different companies.

In multiple case studies each case serves a specific purpose within the overall scope of inquiry (Yin 1994). In particular, Yin (1994) explains that in order to be able to compare findings from multiple cases, the selection of case studies should follow a ‘replication’ logic. *Replication logic* implies treating a series of cases as a series of experiments, with each case serving to confirm or disconfirm the hypothesis. Replication logic aims to show or predict similar results, and explain contrasting results (giving predictable reasons) (Yin 1994). Replication logic is essential to multiple case analysis: it increases the external validity of research (Eisenhardt 1989; Yin 1994) (external validity is discussed in greater detail in Section 4.6).

In this thesis the case study approach focused on a <i>project distributed between at least two locations</i> as a single ‘holistic’ unit of analysis.
--

The next section will explain the criteria for selecting companies and projects for the case studies.



## **4.2 SELECTING THE COMPANIES AND PROJECTS TO BE STUDIED**

The selection of companies and projects to be studied was driven by the main research interest - that is, to study the phenomenon of GD CBD - and followed the replication logic. Multiple case studies in four companies were conducted to increase the external validity of the research (Yin 1994). To correspond with the main interests of the research, case studies that satisfied the following two criteria were selected:

- (1) A CBD project globally distributed between at least two locations of a single organisation (multinational company);
- (2) A project which is successful (according to the measures of success explained in Section 2.6).

Three of the four cases investigated in this research satisfy both these criteria (the deviation of the fourth case from the criteria is explained further in this section).

Naturally, the choice of projects which could be studied was limited, for a number of reasons. First, not many companies were involved in GD CBD in 2001 (at the time of the early stages of this research). Second, not many companies are ready to give access to an external researcher. Therefore, initially the case study selection was limited only by the above-mentioned criteria. However, according to replication logic in a multiple case study approach, I needed to have comparable projects, i.e. similar types and sizes of projects. Thus, in practice, after I obtained access to the first case study in LeCroy, I tried to find more projects with similar characteristics. Following are the secondary requirements for case study selection that were driven by the characteristics of the first case project (LeCroy):

- (a) Type of project – new product development (i.e. innovative projects); interested in long-term collaboration between the distributed teams (as opposed to one-time outsourcing projects that do not plan to have long-term collaboration in the future between the same teams and individuals);

(b) The size of project team is 25-35 people.

The following companies and projects were selected for case studies:

- **Case 1: LeCroy** – development of a CB platform for a new generation of digital oscilloscopes; teams distributed between three locations - Geneva (Switzerland), New York (USA) and Maine (USA);
- **Case 2: SAP** - development of collaborative tools; teams distributed between three locations - Walldorf (Germany), Palo Alto (USA) and Bangalore (India).
- **Case 3: TATA Consultancy Services (TCS)** – development and implementation of a Web-based financial platform for Internet banking. Two related projects were studied: the first project distributed between three locations - Gurgaon (India), Bombay (India) and Zurich (Switzerland); and the second project distributed between Gurgaon (India) and San Francisco (USA). The two projects are related, therefore were analysed together as one ‘embedded’ case study (in this case the two projects are sub-units of analysis) (Yin 1994).
- **Case 4: Baan** – development of an e-Services platform; teams distributed between two locations – Hyderabad (India) and Barneveld (The Netherlands). In January 2002, when data collection in Baan started, the studied project was described by Baan’s contact person as a CBD project, and two teams in Hyderabad and Barneveld were working on the project. However, during data collection, which started from a visit to the Hyderabad office, interviewees reported that the e-Services platform was not CB. Furthermore, in June 2002, when I was supposed to interview people in the Barneveld office, Baan started re-organising its development centres and activities. As a result, in July 2002 development of the e-Services platform in Barneveld was stopped, and later, the whole project was shut down and the Baan facility in Barneveld was closed. Therefore, it is important to note that the Baan case study does not satisfy the case selection criteria, because the studied project (i) is not CB, and (ii) was unsuccessful. Furthermore, it does not fit the unit of analysis, because most of

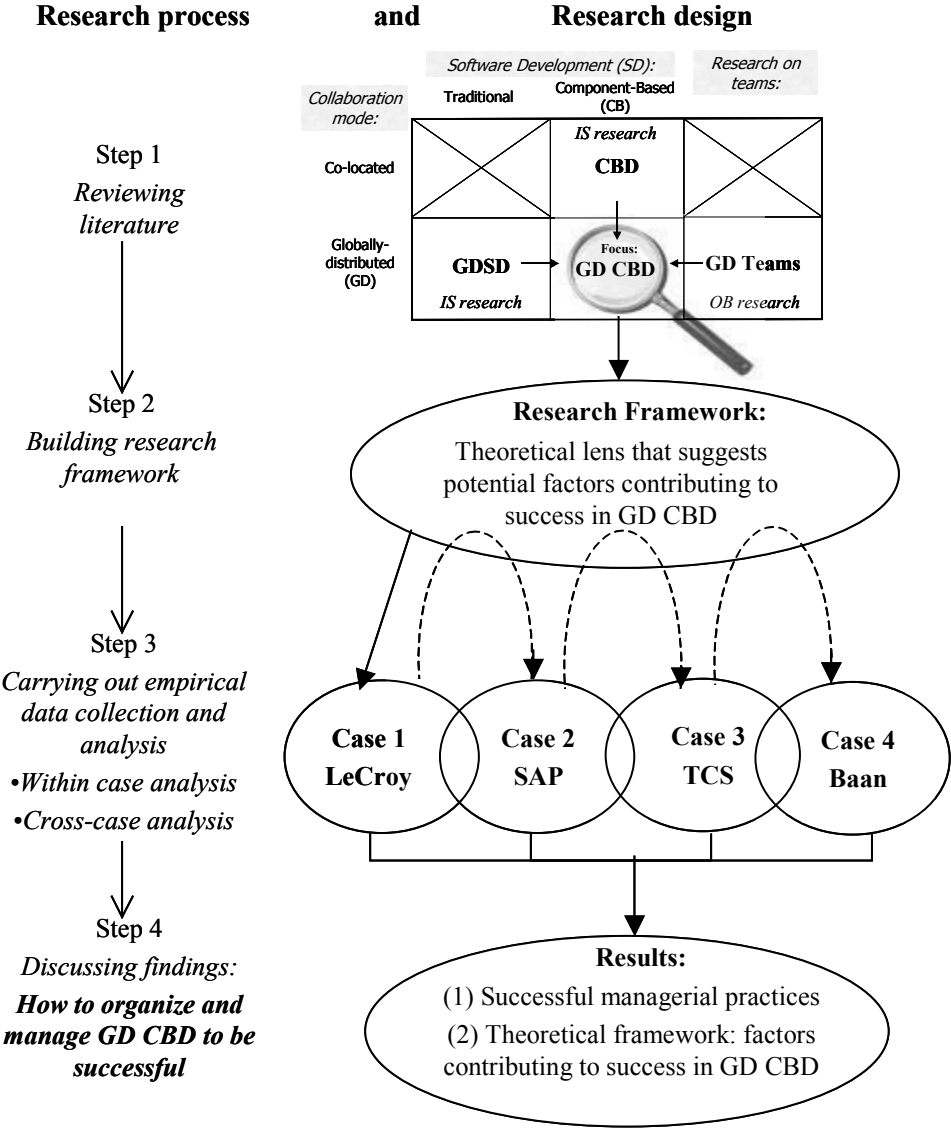
the data was collected in one location (in Hyderabad); data from Barneveld was not available after the project was shut down.

Despite the fact that the Baan case study does not fit either the unit of analysis or the case selection criteria, including it in this thesis gives an opportunity to compare managerial practices from the successful projects of LeCroy, SAP and TCS with practices that were, or, more importantly, *were not* in place in the unsuccessful project of Baan. However, taking into account that the Baan case study is not CB and covers only one of two distributed sites, definite conclusions cannot be drawn from comparing the three successful projects with the unsuccessful project of Baan. The comparison can only suggest some tentative explanations that can be used as a basis for future research.

### **4.3 RESEARCH DESIGN AND PROCESS**

This research was designed in accordance with the replication approach for multiple case studies described by Yin (1994) (see Appendix 1). According to the replication approach, first, each case study is analysed separately, then cross-case conclusions are drawn based on the findings from individual case studies. Figure 16 illustrates the *research design* and corresponding *research process*.

**Figure 16: Research process and research design**



The first step in the research process involved a review of the literature relevant to GD CBD. Based on this literature a theoretical lens was developed (step 2). Then, after preparation which included case study selection, contacting companies and

designing the data collection protocol, the empirical investigation took place (step 3). During the empirical investigation, data analysis was carried out in an iterative way: iterating between data collection and data analysis, as recommended by Eisenhardt (1989). Data analysis continued after empirical data collection was completed. Data analysis included (i) analysis of each case separately (referred to as *within-case analysis*) and (ii) cross-case analysis. Finally, the results from multiple case studies addressing the research question ‘how do companies organise and manage GD CBD to be successful?’ were presented and discussed (step 4). The results include (i) a theoretical framework that identifies factors contributing to success in GD CBD, and (ii) managerial practices that describe how companies organise and manage GD CBD successfully.

The following two sections will elaborate in more detail on the process of empirical investigation and elaborate on methods and techniques used for data collection (Section 4.4), and data analysis and display (Section 4.5).

#### **4.4 DATA COLLECTION: METHODS AND PROCESS**

The empirical investigation included visits to two remote sites per company (in Switzerland and the USA for LeCroy, India and Germany for SAP, India and Switzerland for TCS, and India and, very briefly, The Netherlands for Baan). The duration of on-site visits varied from 1 to 10 days at each site. Evidence was collected from interviews, documentation and observations, as suggested by Yin (1994) and Eisenhardt (1989).

- **Interviews** – semi-structured, open-ended, individual, face-to-face interviews. A semi-structured interview protocol was applied to allow the interviewer to clarify specific issues and follow up with questions on topics related to factors and managerial practices associated with success in GD CBD. It addressed the four potential success factors identified in the theoretical lens (Figure 15). In total, 39 individuals were interviewed (6 at LeCroy; 6 at SAP; 14 at two projects

of TCS; and 13 at Baan)<sup>13</sup>. Interviews lasted from 40 minutes to 1 hour and 30 minutes; they were recorded and fully transcribed. Interviewees were chosen to include (1) counterparts working closely from remote locations, and (2) diverse roles: managers and developers. Managers and developers interviewed at the first location were asked to contact their remote counterparts, introducing me and my research, and to ask their agreement to be interviewed.

- **Direct observations** of meetings (phone and video conferences) and other communications between sites as far as feasible. Observations of working environment and team atmosphere at local offices.
- **Documentary sources** - internal project documents and records, and other internal and external published material, such as press releases, reports and industry data.
- **Informal conversations** with managers and software engineers during lunch breaks and social occasions.

Several rounds of data collection cover a period of time from several months (3 months for TCS, and 5 months for SAP) up to one year (for LeCroy). For all companies except LeCroy this was the time between visits to two remote locations. At LeCroy a second round of data collection took place by phone and emails. Table 4 describes the main steps in the data collection process for each case study.

---

<sup>13</sup> Each case study (Chapters 5-8) includes a through description of the interviewees and other data collection details.

**Table 4: Main steps in data collection process**

<b>Step in data collection:</b>	<b>LeCroy</b>	<b>SAP</b>	<b>TCS</b>	<b>Baan</b>
Initial contact and arrangements	October-November 2001	November-December 2001	November-December 2001	November-December 2001
Visit location 1	Geneva, 1 day in November 2001	Bangalore, 10 days in February 2002	Gurgaon, 10 days in February 2002	Hyderabad 10 days in March 2002
Visit location 2	NY, 5 days in December 2001	Waldorf, 7 days in June 2002	Zurich, 1 day in April 2002	1 day in January 2002 1 day in March 2002
Review of case report for internal validity	Several rounds of emails, July-November 2003	<ul style="list-style-type: none"> <li>• Several rounds of emails September 2002 – August 2004</li> <li>• Feedback session, virtual meeting with SAP interest group June 2004</li> </ul>	By email, February 2005	Personal feedback and by email, November 2004-February 2005
Additional data collection	Phone interviews, November 2003; from Internet sources, winter 2004/2005	By email, summer 2004; from Internet sources, winter 2004/2005	From Internet sources, winter 2004/2005	From Internet sources, winter 2004/2005
Total number of interviews	6	6	14 (for 2 projects)	13

## **4.5 DATA ANALYSIS PROCESS AND INSTRUMENTS**

Data analysis aimed to identify (i) managerial practices perceived by interviewees as contributing to success in GD CBD, and (ii) specific activities that support implementation and facilitate the successful managerial practices. An example of managerial practice would be *designing systematic communications*; and an example of an activity to support the design of systematic communications is *organising regular meetings with all team members from remote locations* (this is one of the activities which helps to design systematic communications in practice).

Data analysis involved qualitative data analysis techniques (described below) suggested by Miles and Huberman (1994), Strauss and Corbin (1998) and Eisenhardt (1989). Analysis proceeded in several steps. It relied on iterative coding of the data using an open-coding technique (Strauss and Corbin 1998), and sorting and refining themes emerging from the data (Miles and Huberman 1994; Strauss and Corbin 1998). The next two sections will elaborate on the data analysis process and describe techniques used for within-case analysis (Section 4.5.1) and cross-case analysis (Section 4.5.2).

### **4.5.1 WITHIN-CASE ANALYSIS**

The data analysis was driven by the research question: *how do companies organise and manage GD CBD to be successful?* This involved, first, identifying managerial practices perceived as contributing to success in GD CBD, and analysing their impact on success. Second, it involved identifying specific activities that helped to implement these managerial practices.

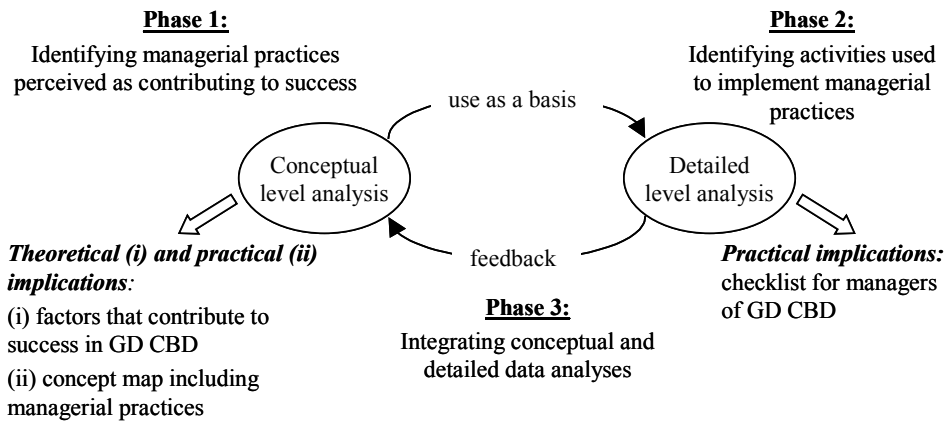
In order to identify these two types of experiences, data analysis was conducted on two different levels:



- To identify managerial practices, data were analysed on a *conceptual level*, focusing on high-level managerial practices and their impact on success, as perceived by the interviewees.
- To identify specific activities that help to implement high-level managerial practices, the data were analysed on a *detailed level*.

Figure 17 illustrates the phases of within-case analysis. Data analysis on two levels increases the internal validity of the research by triangulation of perspectives on the same data set (theory triangulation) (Patton 2001) during the conceptual and the detailed analyses. The findings of the conceptual analysis are of a descriptive nature, while the findings of the detailed analysis are prescriptive (Tsang 1997).

**Figure 17: Phases of within-case analysis**



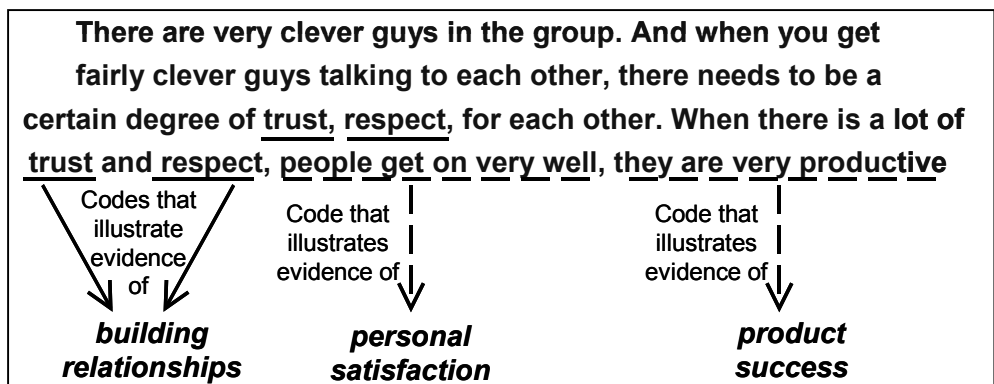
### **Phase 1 - Identifying managerial practices perceived as contributing to success**

This phase consisted of two steps. The first step involved reading through interview transcripts and collected documents and (i) creating a list of practices that were perceived by interviewees as contributing to success in GD CBD, and (ii) marking evidence of success, according to the four dimensions of success:

*product success, personal satisfaction, successful collaboration and bridged gaps* (discussed in Section 2.5).

Chunks of text that are paragraphs or sentences (Strauss and Corbin 1998) describing (i) managerial practices and (ii) evidence of success, were *coded* (assigned tags or labels for later retrieval and categorizing (Miles and Huberman 1994) using an open-coding technique (Strauss and Corbin 1998). *Open-coding* is a process when ‘the investigator identifies potential themes by pulling together real examples from the text’ (Ryan and Bernard 2000): it implies that codes are discovered from the empirical data; that is, new codes are created as new evidence (e.g. issues, themes) emerges from data. The open-coding technique is used when a new phenomenon is investigated, and research focuses on the emergence of theoretical categories from empirical evidence (Strauss and Corbin 1998). Example 1 illustrates how the coding was done, based on a statement from an interview.

**Example 1: Example of codes**



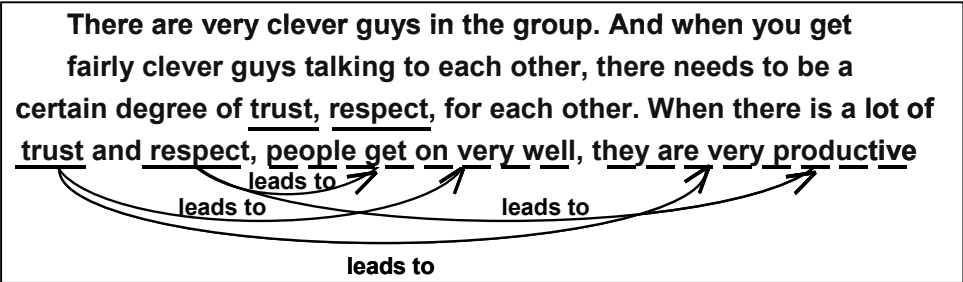
As Example 1 shows, in the above statement the words ‘trust’ and ‘respect’ illustrate building relationships (i.e. social ties): therefore, following open-coding technique, they were marked as codes. Likewise, phrases ‘people get on very well’ and ‘they are very productive’ illustrate success (personal satisfaction and product success, respectively): therefore they were marked as codes. The statement from

the interview used in Example 1 to illustrate coding will be used further in this section to illustrate data analysis techniques applied in this research.

The coding was done in Atlas.ti - Qualitative Data Analysis (QDA) software<sup>14</sup>. The QDA software facilitated the analysis process. In particular, it was used for coding, linking codes and text segments, creating memos, searching, editing and re-organising, and for visual representation of the data and findings (Miles and Huberman 1994; Weitzman 2000).

Activities mentioned by interviewees as having an impact on success were linked to appropriate success measures by creating relationships between appropriate codes in Atlas.ti. Relationships identified are *causal* relationships of types ‘therefore’ ‘lead to’ and ‘in order to’. Example 2 illustrates how causal relationships were established: from interpretation of the statement it follows that *rappport* and *trust* lead to (i) *people getting on very well* and (ii) *being very productive*. Therefore, each of the two codes: ‘trust’ and ‘respect’ was linked to codes ‘people get on very well’ and ‘they are very productive’ and marked as a ‘leads to’ relationship.

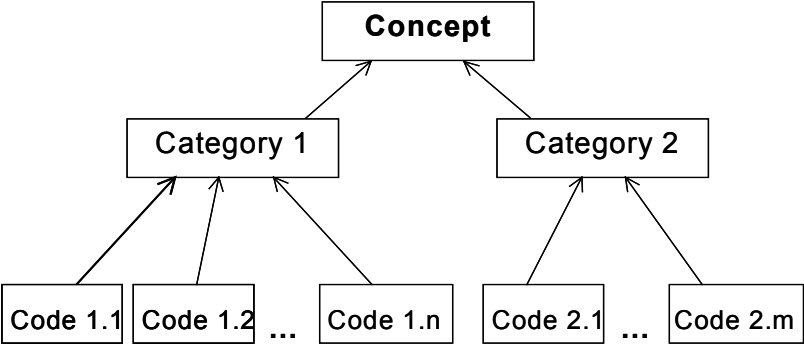
**Example 2: Causal relationships identified in the interview statement**



<sup>14</sup> According to the Atlas.ti web-site (<http://www.atlasti.de/intro.shtml>): ‘ATLAS.ti is a powerful workbench for the qualitative analysis of large bodies of textual, graphical, audio and video data. It offers a variety of tools for accomplishing the tasks associated with any systematic approach to ‘soft’ data--i.e., material which cannot be analyzed by formal, statistical approaches in meaningful ways’.

The second step involved finding conceptual categories and abstractions from the empirical data. This was achieved by grouping codes that share something in common into more abstract higher order concepts (Strauss and Corbin 1998). Categorization of codes makes it possible to reduce the number of units a researcher is working with (Strauss and Corbin 1998) and to clarify the main themes emerging from the data. Figure 18 presents the process through which codes were associated with categories. A bottom-up approach, guided by grounded theory, was used to group codes into categories.

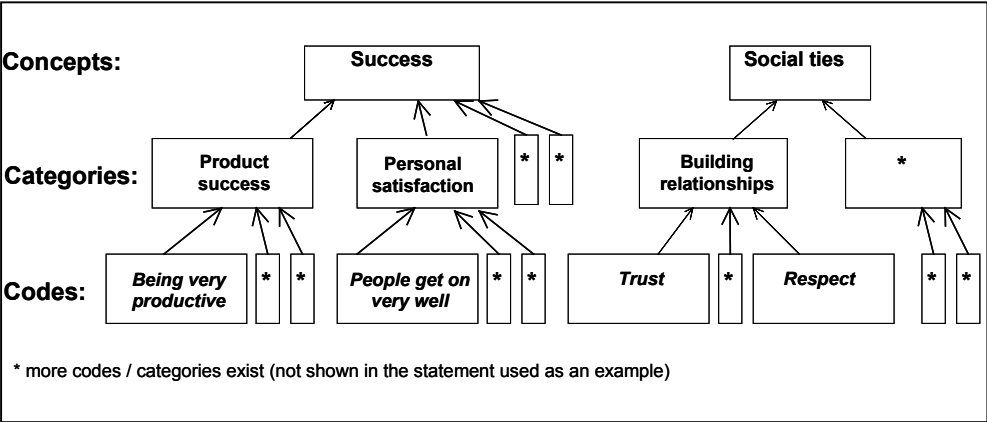
**Figure 18: The data sorting and linking approach**



Codes illustrating (i) managerial practices, and (ii) evidence of success, which were discovered from the empirical data during step 1, were consolidated into broader themes (referred to as *categories*) and categories were classified into concepts. Four potential success factors identified in the theoretical lens (Figure 15) served as concepts. Thus, statements illustrating managerial practices were coded, then *codes* were consolidated into broader themes (*categories*): each *category* represented a different managerial practice. Finally, each *category* representing a managerial practice was connected to one of the four existing *concepts* (potential success factors). If a managerial practice could not be associated with any of existing four factors, a new factor was identified (a new *concept* emerged). For example, *social ties* is a concept; *building relationships* is

one of the categories that represent the concept *social ties*; *trust* and *respect* are two of codes that represent the category *building relationships*. Example 3 illustrates the data sorting and linking approach described in Figure 18 (the example is based on the interview statement used in Examples 1 and 2).

**Example 3: Example of data sorting**

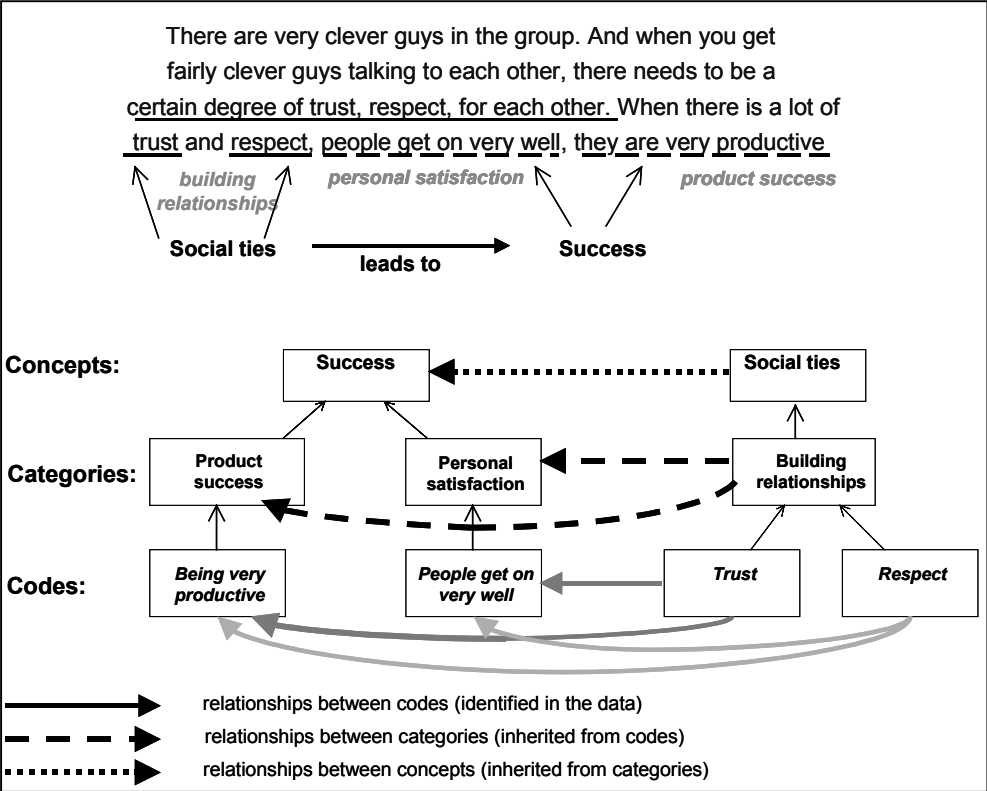


For the LeCroy and SAP cases, identification of themes (categories) emerging from the data, interpretation of selective codes (which seemed to have dual meaning), the consolidation of codes into categories, and the examination of empirical findings against the literature was done together with a senior researcher. The participation of multiple investigators enhances confidence in the findings (Eisenhardt 1989).

Abstraction of conceptual entities (i.e. codes into categories, and categories into concepts) entails the inheritance of relationships between codes (those identified from the empirical data) by more abstract conceptual entities, i.e. categories abstracted from codes inherit relationships identified between the codes. This way, relationships between categories derive from relationships between the codes that categories were consolidated from. Example 4 illustrates how relationships

between concepts (social ties and success) derived from relationships between categories (trust and personal satisfaction / product success), which in turn derived from original relationships between codes identified in the data.

**Example 4: Relationships between concepts / categories inherited from relationships between codes**



The causal relationships between categories representing managerial practices, and categories representing dimensions of success, were used to evaluate the perceived impact of managerial practices on success.

Managerial practices identified during phase 1 served as the basis for detailed data analysis at phase 2.

## **Phase 2 - Identifying activities used to implement managerial practices**

This phase involved reading through the data, and identifying and coding specific activities used to implement the managerial practices identified in phase 1. Relationships between specific activities and managerial practices they implemented are *associative* relationships – activities are associated with managerial practices they help to implement. Relationships are created by linking the appropriate codes in Atlas.ti and are marked as ‘how?’ or ‘by means of’.

## **Phase 3 – Integrating conceptual and detailed data analyses**

During this phase the findings of the conceptual analysis and detailed analysis were integrated. This phase consisted of two steps. The first step included the re-examination of the data that led to the clarification of earlier emerging themes (categories). During this stage categories defined during phase 1 were modified to incorporate the detailed analysis. In particular, new codes created during phase 2 were classified into existing categories or consolidated into new categories. The second step included the preparation of analytical displays linking together all categories, and the discussion of the findings:

### ***Within-case display***

The results of the within-case analysis are displayed in the form of ‘conceptually ordered displays’ suggested by Miles and Huberman (1994): a *conceptually clustered matrix* and a *concept map* (also referred to as *cognitive map*).

A *concept map* displays ‘the person’s representation of concepts about a particular domain, showing relationships among them’ (Miles and Huberman 1994). It is a model ‘typically displayed using boxes and arrows, with the boxes containing themes and arrows representing the relationships among them. Lines can be unidirectional or bi-directional [...]. Relationships can include causality, association, choices, and time, to name a few’ (Ryan and Bernard 2000). Concept maps showing categories (themes) and relations among them are often used to

present the theoretical results of empirically grounded research (Ryan and Bernard 2000).

The concept maps created in this research show managerial practices and dimensions of success (Figures 24, 26 and 33 for the LeCroy, SAP and TCS cases respectively).

A *conceptually clustered matrix* has its rows and columns arranged to bring together items that ‘belong together’. Items in rows and columns are organised in a way that allows for noting relationships between variables (by reading across the rows) and making comparison (by reading down the columns) (Miles and Huberman 1994).

In this research a conceptually clustered matrix consolidates themes that emerged from the empirical evidence as categories (managerial practices), classifies them into concepts (success factors identified in the theoretical lens or those which emerged from the data). This matrix is used to illustrate the contribution of managerial practices to success dimensions (Tables 6, 8 and 19 in the LeCroy, SAP and TCS case studies, respectively). The contribution is assessed based on the frequency of instances in which managerial practices were linked by interviewees with success. The values in the matrix (in all three Tables mentioned above, further referred to as ‘Matrix’) represent the number of instances in which explicit causal relationships (Ryan and Bernard 2000) between managerial practice (row) and category of success (column) was expressed by an interviewee (see Example 2 in Section 4.5.1). The numbers presented in the Matrix give some indication of the strength of each relationship: the higher number the stronger the relationship is.

The Matrix also shows causal relationships between the success factors and categories of success: rows with factors (in grey) show causal relationships ( $\Rightarrow$ ) in intersections with the appropriate category of success. Causal relationships between factors and categories of success are inherited from corresponding



managerial practices, which in turn are inherited from codes (as shown in Example 4, Section 4.5.1). In particular, a causal relationship between a factor and a category of success exists if at least one relationship between any of the corresponding managerial practices and this category of success was identified: in the Matrix these relationships are marked as ‘=>’.

#### **4.5.2 CROSS-CASE ANALYSIS**

Cross-case analysis was guided by replication logic and aimed to compare and explain: (a) similarities, (b) contradictory findings, and (c) complementary findings in the studied companies. A comparison technique was based on listing similarities and differences between the cases (Eisenhardt 1989), and included the comparison of findings and contextual factors across cases.

##### **Cross-case display**

The results of the cross-case analysis are displayed in a *content-analytic summary table*, which focuses primarily on content, without referring to which cases it comes from (Miles and Huberman 1994). In this research a content-analytic summary is represented in the form of a *concept map* that integrates findings across cases (Figure 46).

Furthermore, *conceptually clustered matrices* are used to compare results across cases: managerial practices perceived as important for success (Table 17) and factors contributing to success (Table 21).

The next section will describe the criteria commonly used to assess the quality of empirical research, and will explain how the research presented in this thesis satisfies these criteria.

## 4.6 QUALITY OF THE EMPIRICAL RESEARCH

Tests commonly used to establish the quality of any empirical social research and case study research in particular (as it is one form of such research) are a construct validity test, internal and external validity tests, and a reliability test (Yin 1994).

### ***Construct validity***

Construct validity refers to establishing correct measures for the concepts being studied (Yin 1994). This means that the selected measures (concepts) are measured correctly.

The potential problems of construct validity can be addressed by *data triangulation*, when evidence is collected from multiple sources ‘but aimed at corroborating the *same* fact or phenomenon’ (Yin 1994), and not ‘when you have multiple sources that nevertheless address *different* facts’ (Yin 1994). Gathering evidence from a variety of sources essentially provides ‘multiple measures of the same phenomenon’ (Yin 1994) and ensures ‘stronger substantiation of constructs and hypotheses’ (Eisenhardt 1989).

Other tactics applied in this research to meet the test of construct validity were, first, that key informants reviewed the case study reports, as suggested by Yin (1994). Participant feedback was then incorporated into the final case reports. Second, interpretation of selective codes and examination of empirical findings against the literature for the LeCroy and SAP cases was done together with a senior researcher. Having multiple investigators (triangulation among different evaluators) allows multiple perceptions to clarify meaning and enhances confidence in the findings (Eisenhardt 1989; Pettigew 1990; Patton 2001).

### ***Internal validity***

Internal validity implies ‘establishing a causal relationship, whereby certain conditions are shown to lead to other conditions, as distinguished from spurious relationships’ (Yin 1994).

Two problems associated with internal validity are (1) making inferences (as case study involves inferences, every time event cannot be directly observed), and (2) spurious effects when there are other determinative factors than those identified in the research model.

In this research a number of tactics were used to address these problems and improve internal validity. First, *theory triangulation*, which implies triangulating perspectives on the same data set (Patton 2001), was applied. During within-case analysis the same data set was analysed from different perspectives – on conceptual and detailed levels.

Second, two of the tactics used to meet the test of construct validity: (1) having key participants review and comment on case reports, and (2) participation of multiple investigators in data analysis were applied to meet the test of internal validity as well. To ensure construct validity these two tactics were focused on the understanding and interpretation of the *concepts* studied, while for internal validity they were focused on the understanding and interpretation of the *processes* that can be represented as causal relationships between concepts: one concept (a ‘cause’) leads to another concept (an ‘effect’).

### ***External validity***

External validity implies ‘establishing the domain to which a study’s findings can be generalized’ (Yin 1994).

Using a multiple case study strategy strengthens the generalizability of this research. The design of multiple case studies and cross-case analysis were undertaken according to replication logic, which is the same as that which underlies the use of experiments and allows researchers to generalize from one experiment to another (Yin 1994).

### ***Reliability***

A reliability test aims to minimize the errors and biases in the study. It refers to ‘demonstrating that the operations of a study, such as the data collection

procedures can be repeated, with the same results' (Yin 1994). This implies that if another researcher follows the *same* procedures as applied by a previous researcher for conducting the *same* (and not another) case study, he/she will arrive at the same findings and conclusions (Yin 1994).

In this research a number of tactics were used to ensure consistency in applying procedures for data collection and analysis. First, data collection was guided by an interview protocol designed to capture factors identified in the theoretical lens. This ensured consistency in the areas covered within cases and across cases.

Second, to reduce the likelihood of forgetting or misunderstanding the data, and to allow independent data analysis by other researchers, interviews were taped and transcribed.

Third, use of Atlas.ti qualitative software allowed systematic and consistent analysis of qualitative data (Weitzman 2000): this increases the reliability of research because the procedures can be repeated (Yin 1994).

#### **4.7 CONCLUSIONS: CASE STUDY TEMPLATE**

This chapter has explained various aspects of the research methodology applied in this research. It has provided an explanation of the choice of the qualitative case study methodology, explained the case selection criteria, elaborated on the research design and process, and explained the techniques used for data collection and analysis.

The next four chapters will present the within-case analysis of the individual cases, starting from LeCroy, then SAP, TCS and Baan. The case studies are presented in accordance with the following template (Figure 19). First, the background of the company, team, project and product under study is explained (Section X.1), followed by interview details that contain information about the interviewees and their locations, and by other data collection details (Section X.2). Then, in Section X.3 the analysis and results illustrating how the studied company organises and

manages GD CBD are presented and discussed. In Section X.3.1. a Concept Map illustrating managerial practices perceived as important for success is presented. In the following three sections three types of data presentation (marked as i, ii and iii) are used to support the results reported in the Concept Map and to assess the contribution of managerial practices and potential factors suggested in the theoretical lens to success in GD CBD. (i) In Section X.3.2 the contribution of the managerial practices and potential factors to success is assessed based on the frequency of instances in which managerial practices were linked by interviewees with success. (ii) Section X.3.3. describes managerial practices and presents quotations collected in interviews illustrating these managerial practices and their contribution to success. (iii) Quotations illustrating success achieved are presented in Section X.3.4. Finally, in Section X.4, conclusions summarising case study results are drawn.

**Figure 19: Case study template**

X.1 Background
X.1.1 Background of the Global Organization
X.1.2 Background of the Project and Product Under Study
X.1.3 Background of Software Team
X.1.3.1 Working Experience in a Globally Distributed Environment
X.1.3.2 Organizational Structure of the Software Team
X.2 Data Collected
X.3 How the Company Organises and Manages GD CBD: Analysis and Results
X.3.1 The Concept Map
X.3.2 Factors and Managerial Practices that Contribute to Success: Causal Relationships
X.3.3 Managerial Practices: Description and Evidence
X.3.4 Success in GD CBD: Evidence
X.4 Conclusions

The presentation of results of the Baan case is slightly different from the template used for the LeCroy, SAP and TCS cases. Sections X.1 and X.2 are according to the template presented in Figure 19. Then, Section X.3 examines how Baan organises and manages GDSD, the problems faced and their implications for success. Section X.4 discusses possible impact of the adoption of CBD on the success in the studied project, followed by conclusions (Section X.5).

Taking into account that the Baan case study discusses an unsuccessful project, it was not possible to identify successful managerial practices as was done in the three successful case studies. Instead, the findings from the E-Enterprise project identified several problems faced by the globally distributed E-Enterprise group at Baan, and critical success factors (I asked interviewees about what they considered important to make a globally distributed development successful). Some of these success factors were mentioned because they were lacking in the E-Enterprise project, other factors were mentioned based on the experience interviewees had had in other globally distributed projects of Baan that were successful.

It is important to note that this thesis includes statements made by interviewees, software engineers and managers, many of them non-English speakers. In addition to the ‘ungrammatical’ nature of the English spoken by native speakers, there may therefore be further errors in comparison with standard written English. However, in the vast majority of cases, the sense of the speaker is clear. Where there are slight uncertainties of meaning or minor language errors in the quotes (statements from interviews), I have clarified these in square brackets [ ].



## **CHAPTER 5      CASE STUDY OF LECROY CORPORATION**

*The biggest problem is a people problem: if people from different sites don't have the respect and trust for each other, they don't work well together.*

(Anthony Cake, Chief Software Architect, LeCroy)

### **5.1    BACKGROUND**

#### **5.1.1    BACKGROUND OF LECROY GLOBAL ORGANIZATION**

Founded in 1964 by Walter LeCroy, a physicist, LeCroy Research Systems (in 1980 the name was changed to LeCroy Corporation) was quickly recognized as an innovator in instrumentation. In 1972 the company established an instrument design and production facility in Geneva, Switzerland. In 1976 the corporate headquarters moved to its present location in Chestnut Ridge, New York (NY).

Initially, LeCroy developed technology to capture, measure, and analyse sophisticated electronic signals in a stringent scientific environment. In 1985, the company began transferring this technology to a popular line of general-purpose instruments. Growth in the commercial test and measurement market really took off when the company introduced its first digital storage oscilloscope products. Since that time the core business of LeCroy has been the design and production of oscilloscopes and oscilloscope-like instruments – signal analysers, signals generators and others (see Appendix 2 for general information about oscilloscopes and products of LeCroy Corporation).

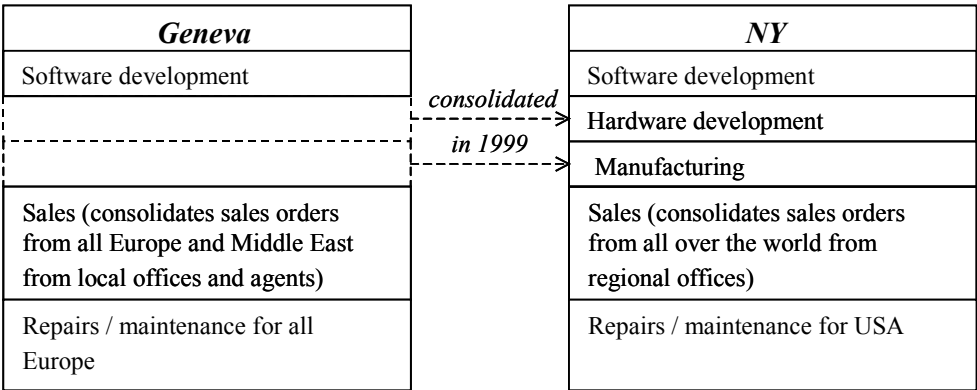
During the last 20 years, LeCroy has opened a number of sales offices in Europe (in France, Italy, Germany, Switzerland and the UK). These offices are responsible for sales in all European countries. There are also offices in Japan, South Korea, China and Singapore (see LeCroy's organizational structure in Appendix 3).



LeCroy now employs more about 400 people worldwide. In 2004 the company reported annual revenues of more than \$120 million<sup>15</sup>.

Three teams – software, hardware and manufacturing – are involved in the production of oscilloscopes. Initially, all three teams were located in New York and Geneva and worked together from these two locations. In 1999 manufacturing and hardware were consolidated in NY. Software development stayed as it initially was, distributed between NY and Geneva. The case study focuses on the software development team, which is globally distributed between New York and Geneva. Figure 20 illustrates the division of responsibilities between the NY and Geneva offices since 1999, after hardware and manufacturing were consolidated in one location.

**Figure 20: Division of responsibilities between NY and Geneva offices**



**5.1.2 BACKGROUND OF THE PROJECT AND PRODUCT UNDER STUDY**

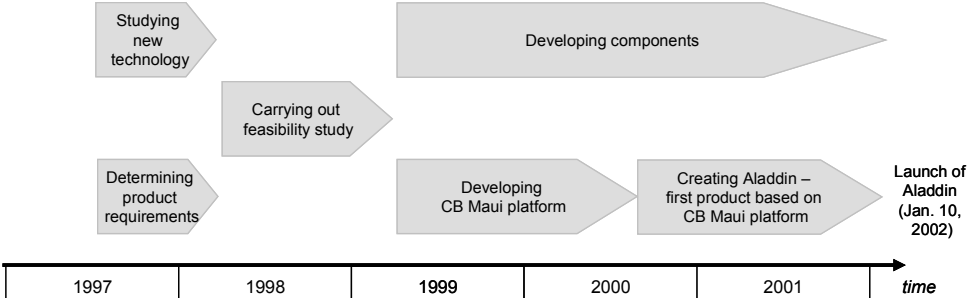
The software for oscilloscopes developed by LeCroy during the period from the 80s until the 90s has grown into a monolithic system. In the first half of 1997 it

<sup>15</sup> From LeCroy Annual Report 2004: this is the latest financial information available

was divided into three modules (operating system, Core software<sup>16</sup> and acquisition system<sup>17</sup>) that that were linked together. Then, while producing scopes based on this modular system, between July 1997 and January 2002 LeCroy developed a Component-Based platform for a new generation of scopes, which is the focus of this research.

The project investigated in this case study concerns the Maui project (‘Maui’ stands for Massively Advanced User Interface). Maui is a software platform for new generations of oscilloscopes and oscilloscope-like instruments based on Windows. This case study covers the development of the Maui platform, and the development of the first products based on the platform. In particular, the focus is on the Aladdin product, the first in the new generation of digital oscilloscopes based on Windows. The launch of Aladdin (officially called WaveMaster) took place on January 10, 2002. Schematically, the major phases of Maui are presented in Figure 21.

**Figure 21: Major chronological phases of the Maui project**



<sup>16</sup> Core contained the functions common to all oscilloscopes (analysis and display capabilities), regardless of the operating systems

<sup>17</sup> Acquisition system is the heart of an oscilloscope; it captures signals. Each scope has a different acquisition system. The acquisition system is the part that changes every time a new scope is produced.

### ***What is Maui?***

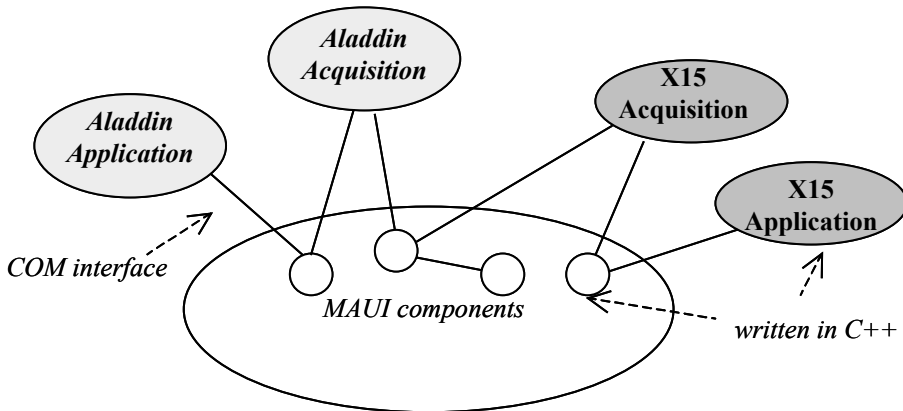
Maui has been called several things. It is an *operating system for scopes*. But basically, it is an application, consisting of a collection of hundreds of components, each of which could have a place in the oscilloscope, or in oscilloscope-like instruments. In other words, Maui is also a *component tool box*: it is a repository of components (there were 508 on December 17<sup>th</sup> 2001); a scope is built by selecting from and integrating these components. ‘That is Maui. However with those components you don’t have a scope’ (Larry Salant, Director of Software Engineering).

### ***How to create a scope in Maui***

A specific oscilloscope product such as Aladdin or X15 can be constructed by integrating the components from Maui with an acquisition system and designing the user interface for a specific application. For example, the Aladdin scope would be built by combining the Aladdin Acquisition system and Aladdin Application with the components selected from the ‘Maui toolbox’. The same would apply for another product called X15: it requires X15 Acquisition and X15 Application together with components from Maui.

The architecture of a product based on Maui consists of large numbers of Maui components (most of them common for all scopes), an Acquisition system and an Application. Figure 22 illustrates schematically Maui product architecture.

**Figure 22: Maui product architecture (schematic)**



Basically, there are four types of components in products based on the Maui architecture. One category is called *processors*, with hundreds of mathematical functions, one component per functionality. A second category is *Graphical User Interface (GUI)*, components that are combined to provide the user interface. The third category of components is the *core components* that allow the systems to work together, and provide the basic instrument capabilities. And finally there are the components that comprise the *acquisition board driver*. These are responsible for controlling the acquisition hardware.

The components are written in C++ and the interfaces between them are in COM. Maui describes these interfaces, they are part of Maui architecture:

I guess, really the root of Maui are these interfaces. There are maybe 30, 40, 50 interfaces which describe how these components talk to each other. That is really the heart of Maui. If you want to make a component for Maui - whether it will be something to display waveforms, to control the front panel, an acquisition system, any of these things – in order to integrate them into the system and to attach to the rest of the system, they have to implement or use one of the Maui interfaces. It is a bunch of standards, and it is a tool kit (Anthony Cake).

As of December 17 2001, Maui architecture contained 508 components. This number had grown to 726 by January 15, 2003 (the number of components on the dates of the first and second round of interviews correspondingly).

### **5.1.3 BACKGROUND OF THE SOFTWARE TEAM**

#### **5.1.3.1 WORKING EXPERIENCE IN A GLOBALLY DISTRIBUTED ENVIRONMENT**

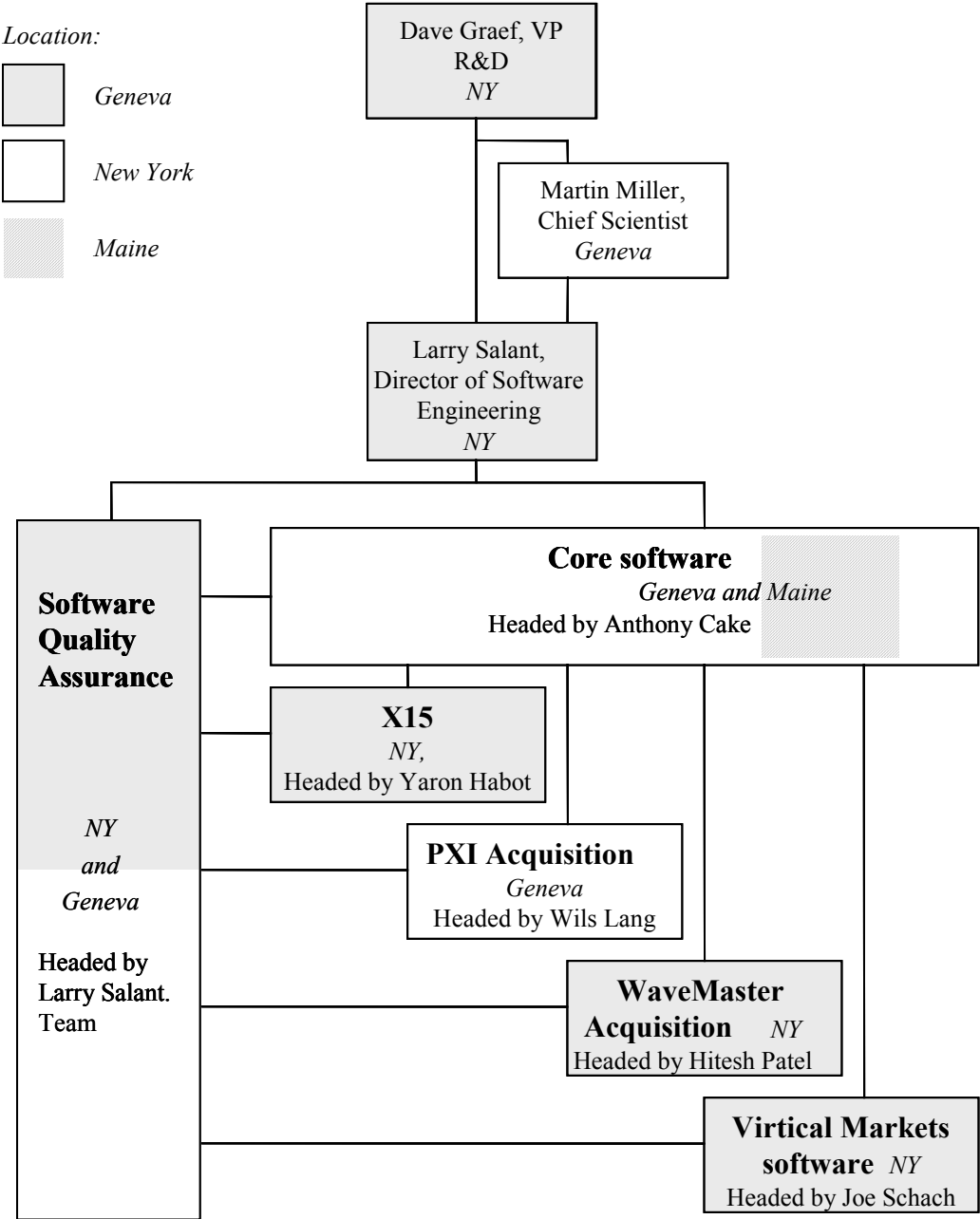
Since the mid 80s, the software for the oscilloscopes has been partly developed in Geneva, and partly in New York. Initially there were about 5-6 people in Geneva and 5-6 people in New York. These two teams interacted frequently. Originally, interactions involved shipping tapes and floppy disks between the two sites. Later, the software team used a 2400-baud modem to interchange files. The interactions progressed as the teams acquired email. Later on, they replaced modems with a Wide Area Network (WAN) connection between the two sites.

In 1999, LeCroy re-examined manufacturing in two very expensive locations. It was not as important to have the software team physically located close to manufacturing. Also, since the software team had developed very good ways of working together over distance (as opposed to the hardware team, which had problems working together over distance), it was decided to leave the software team in Geneva.

#### **5.1.3.2 ORGANIZATIONAL STRUCTURE OF THE SOFTWARE TEAM**

Software development is organised by feature/product function: some features are common in most products and product families (e.g. Core software), other features are developed for one specific product family (e.g. PXI Acquisition). A schematic illustration of an organizational structure of LeCroy software team is presented in Figure 23.

**Figure 23: Organizational structure of LeCroy software team (as of December 2001)**



From a geographical perspective, the software team is distributed between three locations (numbers are correct as of December 2001):

- 1) New York (USA): head office with 13 software engineers
- 2) Geneva (Switzerland): 14 software engineers
- 3) Maine (USA): main software architect (1 person)

Larry Salant is in charge of the NY team and Anthony Cake is responsible for all software engineers in Geneva.

Jon Libby, the main software architect of LeCroy, telecommutes from Maine: ‘Jon is sort of independent, he kind of works on everything, he is one of our architects with Anthony so he basically reports to me but generally Anthony guides what he does because they [Anthony Cake and Jon Libby] are always dealing with architectural issues’ (Larry Salant). Jon had worked in New York for many years and spent a year working in Geneva. He was living in Geneva at the time when the Maui project started. In 1999 his family decided to move back to Maine, where they were originally from. From Maine he carried on the work that he did in Geneva telecommuting. It worked out very well:

Jon is online most of the day with either someone from New York or someone in Geneva talking. Because he is one of the architects of the system, he gets all the guys in Geneva when he wakes up in the morning. They have questions for him and they get on the line with him, and then in the afternoon he has guys in New York who get online (Anthony Cake).

## **5.2 DATA COLLECTED**

Data was collected from a variety of sources: (i) interviews; (ii) internal project and company documents, and external reports and press releases; (iii) direct observations in NY and Geneva offices, i.e. one day in the Geneva office (end of November 2001), and five days in the NY office (December 17<sup>th</sup>-22<sup>nd</sup> 2001); and (iv) informal conversations with managers and software engineers. Table 5 summarizes the names of interviewees, their roles, location (NY or Geneva team),

and details of interviews and other communications for data collection purposes (roles are correct for November 2002).

**Table 5: LeCroy: Interview and data collection details**

<b>Name</b>	<b>Role</b>	<b>Location</b>	<b>Interviews and other communications for data collection purposes</b>
Larry Salant	Director of Software Engineering for LeCroy, responsible for the NY team	NY	<ul style="list-style-type: none"> <li>• Interview at NY office on December 17 2001</li> <li>• Follow-up by email with clarifications and additional information</li> <li>• Review and comments on the draft of the case study report</li> <li>• Phone interview on November 15 2002</li> <li>• Follow-up by email with clarifications and additional information</li> </ul>
Anthony Cake	Chief Software Architect for LeCroy, responsible for Geneva team	Geneva	<ul style="list-style-type: none"> <li>• Pilot interview at Geneva office on November 30 2001</li> <li>• Interview in NY on December 19 2001, during his visit to the NY office</li> </ul>
Gilles Ritter	Software engineer	Geneva	<ul style="list-style-type: none"> <li>• Interview at NY office on December 20 2001, during his stay in NY (in August 2001, he joined the NY team for one year)</li> </ul>
Adrian Cake	Web-master	NY	<ul style="list-style-type: none"> <li>• Interview at NY office on December 20 2001</li> </ul>
Corey Hirsch	VP of Information Systems, Facilities and Security	NY	<ul style="list-style-type: none"> <li>• Several informal conversations in November - December 2001</li> <li>• Review and comments on the draft of the case study report</li> <li>• Phone interview on November 15 2002</li> <li>• Follow-up by email with clarifications and regarding additional internal and external material</li> </ul>
Dave Graef	VP, Chief Technology Officer	NY	<ul style="list-style-type: none"> <li>• Phone interview on November 15 2002</li> <li>• Follow-up by email with clarifications and additional information</li> </ul>



Empirical investigation included several rounds of data collection, which involved visits to LeCroy offices in NY and Geneva, and feedback sessions by phone and email. Data collection covered a period of more than one year, from November 2001 until January 2003, and consisted of the following stages:

- First, initial contact and arrangements for data collection were done with the help of Corey Hirsch. In late November 2001, I visited the LeCroy office in Geneva and conducted a pilot interview with Anthony Cake (manager of the Geneva team) in order to evaluate whether the LeCroy software team met the case selection criteria.
- Then, in December 2001 I visited the LeCroy office in NY where I conducted four interviews, spent several days observing how the software team worked, and collected relevant documents. The interviews took place when the first products based on Maui architecture were due to be released. Interviewees reflected on the period July 1997 – December 2001, which covers the Maui project from the very beginning - feasibility studies and development of the Maui platform - until development of the first product was completed (as illustrated in Figure 21).
- Afterwards, between December 2001 and June 2002, I prepared a draft of the LeCroy case study report (it was also delivered as a White Paper for LeCroy's internal needs). Between July and November 2002, Larry Salant and Corey Hirsch reviewed the draft, and several rounds of emails with comments and clarifications were exchanged during the editing process.
- Finally, in November 2002, I conducted additional phone interviews with Larry Salant, Corey Hirsch and Dave Graef. They were asked to reflect on the progress of the Maui architecture, and the actual (market and financial) performance of products based on the Maui architecture over the year 2002. Following up from this interview, Corey Hirsch provided me with the latest

available financial reports of LeCroy Corporation, and internal and external press releases.

### **5.3 HOW LECROY ORGANISES AND MANAGES GD CBD: ANALYSIS AND RESULTS**

In this section the analysis and results of the LeCroy case study are presented and discussed. First, managerial practices perceived as important in GD CBD are presented (Section 5.3.1). Then, the contribution of these practices to success in GD CBD is assessed and illustrated by empirical evidence from the interviews (Sections 5.3.2-5.3.4).

#### **5.3.1 LECROY CONCEPT MAP**

Data collected in LeCroy was analysed following the approach described in Chapter 4 (Section 4.5). In the LeCroy case, in total 19 managerial practices were perceived by interviewees as important for success in GD CBD. During data analysis these practices were classified into groups that focus on different aspects of management of GD CBD, in accordance with factors suggested in the theoretical lens (Figure 15): (I) *Inter-site coordination*, which focuses on the coordination activities and division of work; (II) *Appropriate tools and technologies*, which describes tools and technologies required in a GD CBD team; (III) *Social ties*, which focuses on people management and social aspects involved in GD CBD; and (IV) *Knowledge sharing*, which focuses on the needs to share knowledge between distributed teams. Furthermore, one more factor emerged from the data, which is (V) *Components management*.

The managerial practices are classified into the five above-mentioned groups (concepts) and presented in the form of the concept map in Figure 24<sup>18</sup>.

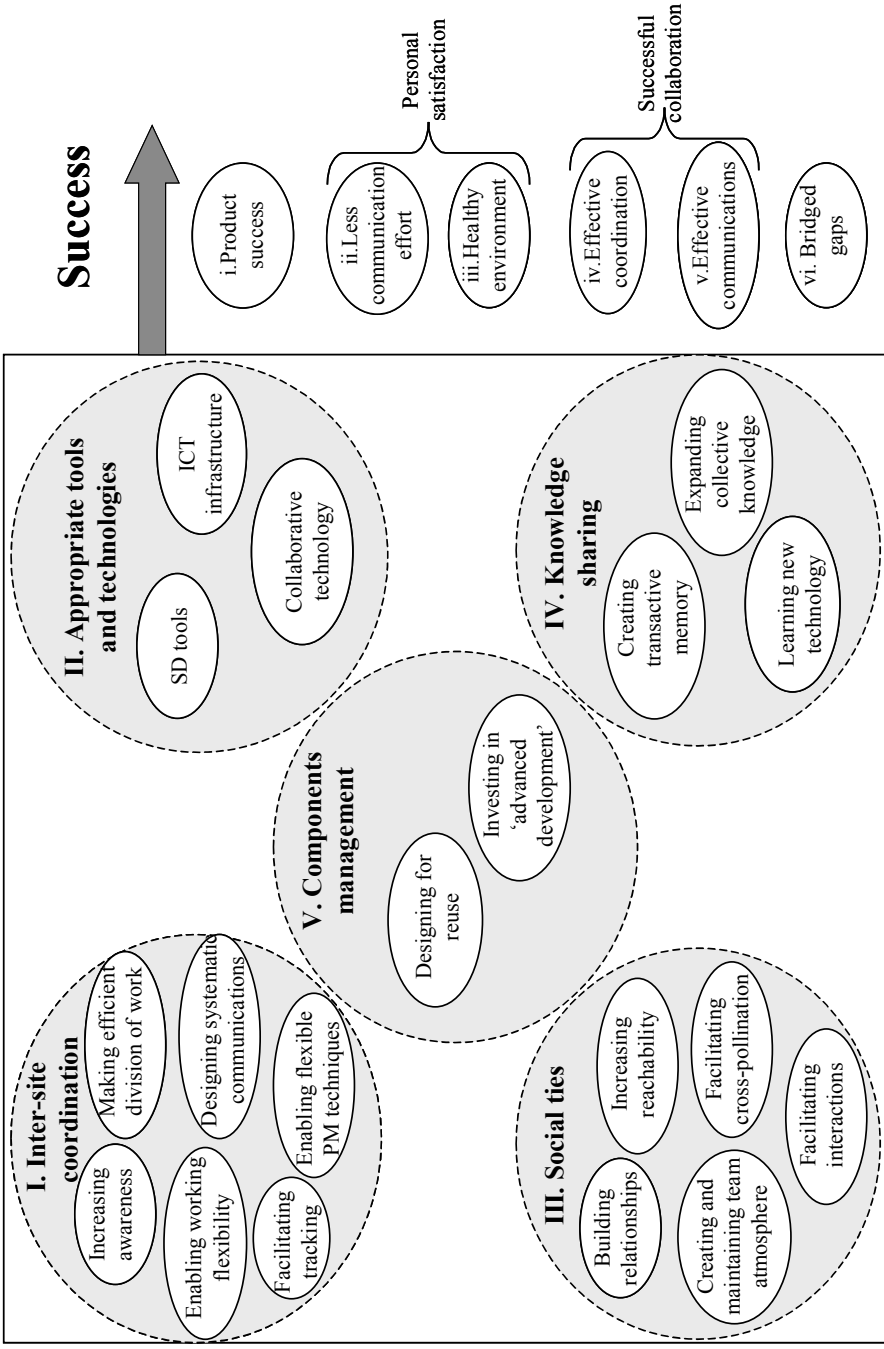
---

<sup>18</sup> The concept mapping approach is explained in Chapter 4, section 4.5.1

In addition to the managerial practices, the LeCroy concept map contains categories representing evidence of success.

In Chapter 2 success was identified as consisting of four categories: *product success*, *personal satisfaction*, *successful collaboration* and *bridged gaps* (Section 2.5). By examining the data from this case two sub-categories of personal satisfaction were identified: *less communication effort* and *healthy environment*. Furthermore, two sub-categories of successful collaboration were identified from the data: *effective coordination* and *effective communications*.

**Figure 24: How LeCroy organises and manages GD CBD to be successful**



In the following sections the contribution of the five factors and corresponding managerial practices to success in GD CBD in LeCroy is illustrated and discussed using three types of data presentation (explained in ‘Within-case display’ in Section 4.5.1 and in Section 4.7).

### **5.3.2 FACTORS AND MANAGERIAL PRACTICES THAT CONTRIBUTE TO SUCCESS: CAUSAL RELATIONSHIPS**

The frequency of instances in which managerial practices were linked by interviewees with success is presented in Table 6<sup>19</sup>. As explained in Section 4.5.1, the numbers presented in the table give some indication of the strength of each relationship: the higher the number the stronger the relationship is.

---

<sup>19</sup> Table 6 is a *conceptually clustered matrix*, explained in Chapter 4, section 4.5.1: it shows the contribution of managerial practices and potential success factors to categories of success

**Table 6: Contribution of managerial practices to success at LeCroy**

Managerial practices	Product success	Personal satisfaction		Success. collabor		Bridged gaps
		Less communication effort	Healthy environment	Effective coordinat.	Effective communic.	
I) Inter-site coordination		=>	=>	=>	=>	=>
1 Increasing awareness			2			
2 Making efficient division of work				1		
3 Enabling working flexibility				2		
4 Facilitating tracking of bugs and development tasks						
5 Enabling flexible PM techniques				1		
6 Designing systematic communications		4	7		7	5
II) Appropriate tools and technologies	=>			=>	=>	=>
7 Software Development tools				1		
8 ICT infrastructure	1					
9 Collaborative technology	1				1	2
III) Social ties	=>	=>	=>	=>	=>	=>
10 Building relationships	2	4	2	2		1
11 Increasing reachability						
12 Creating and maintaining team atmosphere			1			
13 Facilitating interactions		3			1	
14 Facilitating cross-pollination			1	1		
IV) Knowledge sharing	=>	=>	=>	=>	=>	=>
15 Creating transactive memory among team members	1	1	1		1	
16 Expanding collective knowledge of dispersed team		1	2	1		1
17 Learning new technology						
V) Components management	=>					
18 Designing for reuse	1					
19 Investing in 'advanced development'						

From Table 6 it follows that the managerial practices that were most often explicitly connected to success by interviewees are *designing systematic communication* and *building relationships*, and knowledge sharing practices, such as *creating transactive memory* and *expanding collective knowledge*: they contributed to the majority of categories of success. Furthermore, *facilitating interactions* contributes in particular to reducing communication effort.

### **5.3.3 MANAGERIAL PRACTICES: DESCRIPTION AND EVIDENCE**

In this section managerial practices perceived by interviewees as important for success in GD CBD are described and illustrated using quotations made by interviewees. A detailed description of all managerial practices is included in **Glossary of Managerial Practices** (Appendix 4).

#### **(I) Inter-site Coordination in GD CBD: managerial practices**

Following are managerial practices dealing with inter-site coordination in GD CBD:

##### **1. Increasing awareness**

The interviewees indicated that increasing awareness of (i) what is going on in the company and the project, (ii) what everybody is working on in the local team; and (iii) progress made by remote teams, is important for success.

For example, Anthony Cake explained about the philosophy software managers follow: ‘we generally want that everyone knows what everyone else is working on. And if someone is held up because of a particular problem – somebody else may have a solution’.

Furthermore, increasing awareness by providing updated information to ‘other departments (like Marketing and Production) on what we are working on’ (Larry Salant) is considered important as well.

## 2. Making efficient division of work

An efficient division of work is important: it involves the principles that software managers follow (i) to divide work between teams in Geneva and NY, as well as (ii) to divide specific assignments (tasks) and responsibilities between individual team members.

For example:

- To develop experience of new areas (new technology and new products), people who had most experience were chosen: ‘there were guys who wrote fifteen years ago the original code. So they were also the natural guys to work on the next generation, or defining the next generation’ (Anthony Cake). These people developed the basics of the new CB Maui platform.
- Software engineers specialise in different technical domains: ‘each of us know really well one part of the system, so we have kind of specificities, we know better one domain than another one’ (Gilles Ritter).

## 3. Enabling working flexibility

Enabling working flexibility implies providing flexible working conditions in order to accommodate personal circumstances of team members, to make their working environment more convenient and comfortable. The interviewees suggested that working flexibility contributes to success.

The example of Jon Libby, who has been telecommuting from his home in Maine (USA) since early 1999, illustrates working flexibility:

When Jon and his wife decided that they want to move to Maine, we asked him if he wanted to telecommute. I have realized that this month [in December 2001] it is 3 years that he has been telecommuting. He has got a cable modem and he is probably online most of the day with either someone from NY or someone in Geneva talking (Larry Salant).



#### **4. Facilitating tracking of bugs and development tasks**

Tracking includes (i) having a constantly updated status about the stages in fixing a bug, or progress in a task, and (ii) knowing who is responsible for fixing the bug, or completing the task.

Tracking of bugs is important for developers as well as for people in sales offices. At LeCroy the in-house developed tool BugBase is used for tracking bugs and development tasks:

Everyone has access to BugBase, also all our sales offices, in Japan, for instance, they have a copy of it. They can enter the bugs and they can look at the status. And what happens is, as a bug gets fixed, the one who entered the bug gets notified that it was fixed. And every time one of the engineers changes the bug, as a manager I get notified that they updated it, and so I can see how they diagnosed it. So for management of bugs BugBase invaluable (Larry Salant).

Furthermore, BugBase is used for tracking development tasks: ‘sometimes we put into BugBase tasks for people, just because it is convenient later to track them’ (Larry Salant).

#### **5. Enabling flexible Project Management (PM) techniques**

Flexible PM techniques are necessary to accommodate everyday dynamics: ‘the problem we find in huge projects in particular - there are so many dynamics – things dynamically changing on any given day. If you try to fully maintain the project at micro level, it would be a full-time job for someone’ (Larry Salant).

Therefore, a flexible PM technique includes:

- On a macro level: planning of major project activities (milestones)
- On a micro level: flexible and not too detailed planning

## **6. Designing systematic communications**

Systematic communications are considered by interviewees as important for success. This practice includes organising frequent communications and designing rules aiming to make communications more effective, in particular:

- Scheduling systematic and frequent communications, such as regular teleconferences between software managers in NY and Geneva; a transatlantic videoconference with all team members every couple of months
- Communicating directly to reach an appropriate person (i.e. no hierarchy in communications).
- Improving style and content of communications to achieve better understanding (and prevent conflict and misunderstanding) between remote counterparts.

For example:

I have a lot of experience working with a lot of foreign cultures. In some cultures if you are on the phone explaining something to somebody and they don't understand it – they still say 'I understand'. So the way I try to ensure that the information was received correctly is through a very detailed process of describing the issue. For example I say, 'open this Web link. What do you see?' So it is very specific, very detailed (Adrian Cake).

## **(II) Appropriate Tools and Technologies in GD CBD: managerial practices**

Managerial practices related to tools and technologies important in GD CBD are as follows:

### **7. Software Development (SD) tools**

Software Development (SD) tools include tools for the development and management of components, configuration and version management tools, and tools for testing and tracking bugs. In order to support CBD in a globally distributed environment SD tools need to provide the following capabilities.

---

**Automated management of interdependencies between components and related files**

Managing interdependencies between components is not a problem as long as the number of components is small. In this case the dependencies could be modelled and understood visually. However, when the number of components becomes hundreds, visual understanding is no longer an option. As Anthony Cake explained:

Imagine building one DLL in one project under Visual Studio. It is very easy to do. Building 2 or 3 project DLLs that depend on each other is also fairly easy to do. Building 300 or 500 of these things is impossible.

To this end, the LeCroy software team developed an in-house tool called COMProjMgr. COMProjMgr knows the dependencies between all the files in Maui (of which there were 5,000–6,000 in end of December 2001), and between the various components, and manages the entire project.

---

**Rapid update of changes**

There are many dynamics in the development environment: every day new components are developed, and existing components are modified. Components and files are inter-related, thus every new component and modification requires changes in the whole environment. In order to accommodate rapid changes in the development environment and ensure that everybody is working with the latest versions of files and components, the LeCroy software team programmed four times a day a build of components. The building does not apply to everything, as this would take too long. ‘Right now [end of December 2001] building everything takes about six

---

---

hours, even on a high-powered machine’ (Anthony Cake). Therefore COMProjMgr builds only those files that have been modified or added, and those that depend on them:

What COMProjMgr will do is if one of these files changes, it knows the dependencies about everything from everything else. And it will go through the old build just looking for things that need to be built (Anthony Cake).

---

**Automated testing of components**

An in-house developed tool called SoftwareTestHarness is used for testing components. It runs automatic tests every day:

What it does, it shows you all the LeCroy developed components in your system, and you could say ‘run a test for all of them’ or ‘run the test for any one of them’ (Larry Salant).

Each component LeCroy develops has interfaces that are standard for the component: one for a basic self-test, and one for an advanced self-test. There are special test components used for testing of other (functional) components. They typically contain ‘a whole bunch of test cases’ needed to make sure that the functionality of the tested component is correct. ‘We can test each component by itself in SoftwareTestHarness and that runs every single day automatically’ (Larry Salant).

---

<b>Standardization of tools and methods across locations</b>	<p>Everyone working with Maui uses the same tools and methods:</p> <p>All are identical, absolutely identical. We have one Version Control System [VCS], at least for Maui, which is located in Geneva: it is on the network, so everyone can get to it. The Lotus Notes system we use is on servers in NY and in Geneva. And they are replicated, so they are identical essentially. Everything is the same. Everyone working with Maui uses the same tools (Anthony Cake).</p>
<b>Centralisation of tools</b>	<p>Centralisation of tools in one location ensures one single environment for all remote locations. For the LeCroy software team, there are no ‘local’ tools as such – all tools are located at one central place. For example:</p> <p>VCS – Perforce – exists in Geneva and guys access it here [in NY] the same way over WAN, so the only difference there is: from here it takes a little longer to access it, speed is slower. It doesn’t matter where you are in the world, you still can access the same single VCS (Anthony Cake).</p>
<b>Creating a Guide that explains how to use tools and methods</b>	<p>Maui Software Developer's Guide lists the tools used (Lotus Notes, Visual Studio, Perforce, ComProjMgr, Rational Rose, BugBase, SoftwareTestHarness) and explains how to create and debug components in Maui using these tools. This Guide is invaluable for new employees, and staff moved from previous products and starting work with Maui (during a transition period).</p>

---

**Developing tools in-house**

The strategy regarding SD tools that LeCroy software managers follow is building tools in-house, if the required tools are not available on the market. Anthony Cake and Larry Salant both have the same opinion:

Whenever we need a tool, we do try to buy it, but most of the time we don't find a proper solution. Then we made our own, and this goes for most of the tools that we have.

Of the main four SD tools, Perforce is a commercial tool, and the other three - COMProjMgr, SoftwareTestHarness and BugBase (discussed above) – are all tools developed in-house by the LeCroy software team.

---

---

**8. ICT infrastructure**

---

Interviewees stressed the importance of ICT infrastructure for success: ‘no firm trying to execute GD CBD successfully can do so without the right infrastructure’ (Corey Hirsch).

An ICT infrastructure enables connection between all remote sites. It includes Internet, WAN, server and applications pool, how resource shares are set up (i.e. sharing of databases, server, project repository), conferencing tools, and network speed and bandwidth. Furthermore, it includes capabilities aiming to support security requirements, such as firewalls and access rights.

In order to succeed in a globally distributed environment the ICT infrastructure needs to support the following:

<b>Quick access to the network</b>	Quick access to the network is required from all remote locations (in the office, and for those working from home).
<b>Shared databases</b>	Having one central database accessible over WAN from remote locations ensures that everyone is working with the latest versions of files and components:  I don't have to build every component locally. If someone changes the hardcopy component and they put it back – it will be rebuilt on the server and then in the morning I can import that component and just use it (Larry Salant).
<b>Web access and constant replication of databases</b>	Web access and constant replication of databases (over the Web) are required to provide updated information/data and allow tracking. LeCroy engineers have project databases based on Lotus Notes. As Larry Salant explained:  Because we are working at separate locations and Lotus Notes replicates databases, it is very good for us. The big databases are local to Geneva and here [NY] and they get replicated constantly over the Web.

---

## 9. Collaborative technology

The following are collaborative technologies used by LeCroy team to collaborate successfully over distance:

---

<b>Online chat</b>	<p>Every member of the software development group appears on the list of MSN Messenger. This tool enables real-time remote contact:</p> <p style="padding-left: 40px;">During the day if you have a question or you need somebody's help, largely you use online chat. It is immediate, it does not matter where they are in the world – whether they are in the next cubical or whether they are in the next country, they use that system (Anthony Cake).</p>
<b>Phone and teleconferencing</b>	<p>If real-time collaborative tasks require more than a couple of lines of response, team members tend to communicate by phone: 'generally if it is more than a couple of lines of response, then we'll pick up a phone, and talk to each another' (Anthony Cake).</p>
<b>Application Sharing</b>	<p>The LeCroy software team uses the Net Meeting Application Sharing Tool (AST) for real-time collaboration, both collocated and remote collaboration. It allows developers to see what is on the screen of a remote computer, and to share and take over control. Software developers make extensive use of the tool for code reviews. Larry Salant observed:</p> <p style="padding-left: 40px;">I have even seen it within this building, two guys in almost the next cubical to each other doing a code review: sitting next to each other, but they are sitting at their desks and looking at their own screen, working through the code. So, it is actually an interesting tool, and people are used to doing code reviews across the ocean or up to Maine [with Jon Libby].</p> <p>Larry Salant and Anthony Cake also use the AST frequently when designing a new feature or user interface: 'we have been working in Visual Studio when laying out a dialog for a product via AST when Anthony will be in Geneva and I'll be</p>

---



---

here [in NY],’ (Larry Salant).

AST is used for taking control of a computer mainly when somebody needs help with debugging. For example:

If someone has a problem in Geneva and would like to work with me on finding the bug in the code, we use AST to go through the code together [NY and Geneva] while discussing the bug over the phone (Gilles Ritter).

Typically in such situations developers use AST to see what is happening on the computers, and at the same time they use the phone or voice chat capability of AST to discuss the problem.

---

**Videoconference** Since about one year before the launch of the Aladdin system (from early 2001), software managers have been Video Conferencing (VC) at least once a week to discuss progress and other issues. Furthermore, VC is used (i) for meetings with a remote team (e.g. when Anthony Cake visiting NY office, he holds meetings with his team in Geneva via VC); and (ii) for meetings with all developers from both locations:

Every once in a while, more recently as NY guys also started working with Maui, we have trans-Atlantic videoconferences with all the software guys in NY and Geneva (Anthony Cake).

---

**Email** Email supports low priority tasks and issues, and tasks that cannot be completed in real-time because of time-zone differences. ‘Stuff that doesn’t need an immediate answer or things that happen outside of the overlapping time period, that all happens by email.’ (Anthony Cake).

---

**Intranet** The LeCroy team has access to its own Intranet environment where internal documents and other relevant information are posted.

---

### **(III) Social Ties in GD CBD: managerial practices**

According to the opinions of the interviewees, rapport and trust contribute to success, as illustrated by the following quotations:

#### ***Contribution of Trust***

It makes a big difference, when the guys know each other. And more importantly – when the guys trust each other and they know what the others' capabilities are. I think that makes a huge difference. It is because there are very clever guys in the group. And when you get fairly clever guys talking to each other, there needs to be certain degree of trust, I guess respect is may be a better word, for each other. And where that is lacking, there is really a communication problem. But when there is a lot of trust and respect, people get on very well, they are very productive (Anthony Cake).

#### ***Contribution of Rapport***

We found over the years that whenever people had worked face-to-face, or even if it was only for a few days, the fact that you could put someone's face to it, made it that much easier for someone to pick up the phone and ask the question, than if it was just a name that you heard (Larry Salant).

Following are the managerial practices that focus on social aspects and facilitate rapport and trust between remote counterparts.

### **10. Building relationships**

Building relationships involves building rapport and trust between remote team members: it is considered by interviewees very important for success. The following quotes illustrate the importance LeCroy managers give to building relationships:

- 'We all got together in the mountains of France and it was a real fun week. It had two purposes: one was to teach us all this new technology [Microsoft COM]. The other, which was equally important, if not more

important, was to try to build relationships between people' (Larry Salant).

- 'The biggest problem is a people problem, or people from different sites, it happened, do not respect and trust each other, they don't work well together. But in most of cases that is not really an issue any more' (Anthony Cake).

---

## **11. Increasing reachability**

---

Increasing reachability implies making it easier to reach the right people at a remote location, in particular:

- to know whom to contact, i.e. who is the person who has knowledge (of a certain domain or issue);
- to know who is available, i.e. if the person is in the office on the given day or time.

For example, as everyone appears on the MSN Messenger list, this gives an indication to others, specifically in the remote locations, about who is at work (logged in MSN Messenger), and if the person is at his/her desk or away (status changed to 'away').

---

## **12. Creating and maintaining team atmosphere**

---

Creating and maintaining a team atmosphere implies making sure that all are 'plugged' into the project/company. It is important, in particular for the remote team in Geneva:

What happened in Geneva is that among the guys there is a natural feeling that they are kind of unplugged from the rest of the company. Because it is an outpost! In order to handle that we organise regular meetings to let people know what is going on in the company, what everyone else is working on. It is a big help. Every several months we have a transatlantic videoconference with the software guys in NY and Geneva. It helps everyone, I think, to feel we are working as a team and that they are part of the LeCroy team (Anthony Cake).

### 13. Facilitating interactions

Facilitating interactions between people at remote locations is important. It includes (i) facilitating personal face-to-face interactions and (ii) organising regular and frequent interactions over distance.

LeCroy software managers try to facilitate interactions and create relationships between remote counterparts: ‘we try to make sure they interact, we increase the possibility that they really get to know each other (Anthony Cake).

For example, meetings in person are considered important:

Meeting and getting to know each other has got a lot to do with trust and respect. In fact, I would say that most valuable time spent in this respect is probably in the local bar than in the meeting room. Because getting to know someone happens over a few beers. And that develops into the professional [area]. I think that’s sort of important thing, very important thing. That was the idea behind the conference in the Alps, to get people in an environment where there was plenty time for that. It was pretty important (Anthony Cake).

### 14. Facilitating cross-pollination

Cross-pollination implies that people from the one group spend significant amounts of time in the other group (other location) and vice versa.

One of the interviewees emphasized the importance of cross-pollination by giving an example of unsuccessful collaboration of the LeCroy hardware team. Initially the hardware team was distributed between NY and Geneva, the same as the software team:

I think, part of the problem was – there was no kind of cross-pollination. There was nobody from the NY group who spent a significant amount of time in the Geneva group or vice versa. So there were already two separate groups. How to explain, they just didn’t get on. Really didn’t have any respect for each other (anonymous, as requested by interviewee).

In the software team, one of the advantages was that a couple of members of the Geneva software group originally worked in New York. Anthony Cake started in NY in 1986, and only later moved to Geneva. Another senior person – Martin Miller, the chief scientist currently based in Geneva - worked in New York for many years (he has been at the company since the late 1970s). Anthony Cake expressed his viewpoint: ‘to take people with experience, I think, working in the group, and then move them into another group, is a good way to seed the other group, to make sure that everything works together’.

#### **(IV) Knowledge Sharing in GD CBD: managerial practices**

Interviewees consider knowledge sharing as contributing to success, in particular, building up collective knowledge through shared experiences, and creating transactive memory among team members at dispersed locations.

In LeCroy team members had a history of working together, and some of the dispersed team members had an opportunity to meet in person: therefore at LeCroy global software team transactive memory and collective knowledge were developed to some extent before the case project started and were facilitated throughout the project.

Following are managerial practices seen as important for knowledge sharing between remote team members, supported by quotations from interviews.

#### **15. Creating transactive memory among dispersed team members**

Creating transactive memory among team members located in NY and Geneva and Maine is considered important for success.

In LeCroy a number of activities that facilitate interactions among dispersed team members were organised through which team members could get to know each other and further facilitate creation of transactive memory. These activities

included an introductory course for Microsoft COM combined with a team-building exercise, where all team members met in one location, and also frequent visits of managers to remote locations.

The following quote illustrate the existence of transactive memory at the studied team:

- ‘When a problem occurs it is important for the team, instead of finding the bug, to find quickly who knows best about the failing component’ (Gilles Ritter).

## **16. Expanding collective knowledge of the dispersed team**

Expanding collective knowledge of the dispersed team is important for success. This practice includes learning about the national culture of remote counterparts, sharing knowledge of the overall product (beyond a specific area an individual team member is working on) and developing common technical knowledge.

Development of the Maui platform started in Geneva where the basics of the platform were developed; only later did the team in NY start working on Maui. Gilles Ritter, who was involved in the Maui project from the very beginning, explained how knowledge sharing about the Maui platform was organised to ensure collective technical knowledge and common understanding of the evolving product:

Initially only a few people started in NY and they had always a lot of questions regarding the new platform. So they were always in contact from NY to Geneva. And when more and more people in NY started to work on the new platform, it was decided for me to come over here [to NY] for one year to facilitate the contact for everyone new in the new platform. [...] I know all the basics, the background of the platform. So, that’s why I am here for one year to kind of teach all the other co-workers how to develop using the same tools.

To expand collective knowledge of the dispersed team members LeCroy managers facilitate sharing of experiences between the teams, as illustrated by the following quote:

I am back and forth all the time, and Anthony as well. But occasionally, we do have people coming from Geneva here or from here going to Geneva for a week or two and we even have a few cases where we put someone over, we have one guy right now who is spending a year here from Geneva. And that is really useful sharing experiences and stuff (Larry Salant).

## 17. Learning new technology

For LeCroy software engineers the new Microsoft COM technology and CBD methodology were very different from the approaches they used to develop software for earlier oscilloscopes. Therefore, one of dilemmas LeCroy faced while developing the CB Maui platform was how to move people onto the Maui project so that they could develop in Maui and, hopefully, be as productive as they were with the old system<sup>20</sup>. Thus, learning new technology was organised in several

---

<sup>20</sup> Anthony Cake explained:

It's an interesting or it's a difficult step for a developer to make, when you were the master of your environment for such a long time, and you understood the entire system (and it is - we are talking about half a million lines of code). These guys knew this stuff [the old system], this was their world for 10-15 years, and all of a sudden someone says 'OK, forget all that, we are going to go to this new place which is completely different'. And it is using some standards by Microsoft, that we didn't create and that's not perfect but we have to live with them. And, everything that they were used to day-to-day - changed. Some guys accepted that very, very quickly and some guys were up-and-running, maybe climbing the learning curve within a few weeks. Other guys, they took longer. Somebody from the original senior guys are still not really up to speed in this new environment - they never will be as productive as they were on the old stuff. So the younger guys find it a little easier, they came up to speed literally in weeks.

steps. First, the introduction of the Microsoft COM technology was organised, when all software engineers had an overview and some background about its principles and development methodology. The second step involved learning how to work with applications based on Microsoft COM technology. Finally, after the Maui platform was developed by a small group of experts, all software engineers were taught about what the Maui and how to develop a product (oscilloscope) in Maui.

Furthermore, after the Maui platform was developed, a Guide that describes the environment and tools used to develop products in Maui was created. The Guide served as a reference framework for everybody and facilitated learning of the new platform:

The Maui Software Developers Guide is a kind of getting started guide for new engineers coming on board with Maui. Because one of the problems we had is that our old system was a heavily embedded system based on embedded operating systems and embedded compilers. And moving those developers into Maui and using tools like Visual C, things like Rational Rose for the UML diagrams, means that everything that we used and lived in for years changed. So this Guide, this Bible is explaining how to move into this new development environment (Anthony Cake).

#### **(V) Components Management in GD CBD: managerial practices**

In addition to the four factors suggested in the theoretical lens, the *components management* emerged from the data as a factor contributing to success. Following are managerial practices seen as important for ensuring the successful components management in GD CBD.

#### **18. Designing for reuse**

For LeCroy this practice aims to increase reuse of software components across a number of products in the long term. This involves analysis and long term planning for future products and product families, and making strategic decisions



about the granularity level of components. The need to facilitate reuse through design derives from the major goal of LeCroy software managers:

- ‘We wanted to have a system that really is Object Oriented and reusable and modular and all these good words [...]. We developed this architecture [Maui] to be built on for years in the future’ (Anthony Cake).
- ‘The whole idea is that we can take the bunch of different components and create a different instrument, within weeks is kind of optimistic, but within a few months rather than in a few years’ (Larry Salant).

### **19. Investing in ‘advanced development’**

The development of the Maui platform was treated at LeCroy not as a typical product development project where product requirements are defined in the very beginning, but as a research project: ‘we were really trying to determine if we can build a product on it [Microsoft COM] and doing some essentially pure research - what people would call ‘advanced development’ (Larry Salant).

*Advanced development* included learning about available technologies, and conducting a feasibility study aiming to test whether or not a ‘proof of concept’ for the product can be achieved by applying available technology(ies):

When Maui project started, we didn't really have a product in mind, not in the sense of the product that you can ship. But we knew that we wanted to use this [Maui platform] on several products that would be defined in the future (Anthony Cake).

#### **5.3.4 SUCCESS IN GD CBD: EVIDENCE**

This section presents evidence collected in interviews about the success achieved in the studied case. The evidence (quotations from interviews) is presented according to the categories of success illustrated in the LeCroy Concept Map (Figure 24).

## **i Product Success**

The Maui project was highly successful:

- LeCroy's WaveMaster 8600 was announced as Product of the Year 2002 by END magazine (among ten best products for test and measurement purposes)<sup>21</sup>.
- The Maui CB architecture (platform) served as a basis for future products:

We began shipping both the WaveMaster 8300 and 8500 to customers in March, 2002. At the same time we also began shipping a Disk Drive Analyzer (DDA), which is based on the WaveMaster 8500 (Larry Salant).

- In January of 2003, LeCroy launched the WavePro 7000 series of scopes (7000, 7100, and 7300), which is also based on Maui.
- Due to the Maui architecture LeCroy successfully partnered with three different commercial software companies during 2002 to further extend the analysis capabilities of LeCroy products.

## **Personal Satisfaction**

### **ii. Healthy environment**

The job here is very demanding and challenging. I think that those who stay onboard are the engineers who share the same goal: to work on complex problems in cutting edge technologies. I think that that the fact we share this goal helps us to communicate well (Gilles Ritter).

### **iii. Less communication effort**

- ‘We use MSN messenger from Microsoft - every member of the software development group, they appear on the list. So for having a chat with someone, wherever they may be in the world in the given

---

<sup>21</sup><http://www.e->

[insite.net/ednmag/index.asp?layout=article&articleid=CA263115&pubdate=12/12/2002](http://www.insite.net/ednmag/index.asp?layout=article&articleid=CA263115&pubdate=12/12/2002)

time, you just need to double click on their name and start typing a line' (Anthony Cake).

- 'In Geneva, all senior guys speak English very well. Some of the junior guys speak English purely. So what we have done at their request, we paid English lessons for them. But locally they speak French. When I communicate with them in English, it is very rare that I cannot communicate my ideas or issues or so on' (Larry Salant).
- Gilles Ritter explained about his experience of working from Geneva with Jon Libby located in Maine: 'I think because we started to know each other better, we know each other's feelings better, so now even before asking him a question I know how he is going to start to think'.

## Successful Collaboration

### iv. Effective coordination

- 'Basically when we started the platform in Geneva we were only a few who developed the basement of the new platform [...]. And the other guys, the other workers who joined us after and had to learn how the platform works, now know who of the first guys knows well which parts, then go to ask questions. And sometimes, if it is not the right person, I'll just tell him to ask another guy who knows better than me, and this is how it works' (Gilles Ritter).
- Having standard and centralized tools helps to make coordination more effective and efficient. For example, there is no need to build every component locally: all components are built on the central server. Furthermore, programming building of components four times a day allows the use of time-zone differences to work around-the-clock (see practice 'ICT infrastructure' in the previous section).

### v. Effective communications

Gilles Ritter explained about his experience of working with remote counterparts:

For example when I control his machine, it doesn't respond as fast as on my computer. So it is a technical delay in terms of seconds, but the understanding is absolutely identical remotely or just on site.

## vi Bridged geographical, time-zone and cultural gaps

*Geographical distance* is not perceived as a problem:

For LeCroy software team, geographical distance causes limited inconvenience, i.e. in extreme situations when physical presence is required: ‘for an important meeting, people get on the plane and fly over for a meeting, but that is an extreme’ (Anthony Cake). But on a regular basis, team members communicate remotely using different types of communication media.

*Time-zone differences* are not perceived as a problem:

Time differences are not perceived as a problem, rather as an advantage: ‘we use the fact that we are not working together to allow us to work around-the-clock’ (Gilles Ritter). Anthony Cake explained:

Generally it doesn’t really matter, it is not a big advantage, not a big disadvantage. I would not say that time differences are a disadvantage, and close to a release or big milestone they can be a big advantage. Because problems, bugs fixes, can be passed on from time-zone to time-zone.

There is a 6 hours’ time difference between the USA East coast<sup>22</sup> (UTC –5) and Switzerland (UTC +1). Despite this 6 hours difference, ‘generally we have quite an overlap. Because, the first guy that starts working in Geneva is in the office at about 6 am. And in times when we are close to getting a product out, or big milestone, they are there [in the office in Geneva] until midnight, so we get only a few hours when we are not overlapping somewhere’ (Anthony Cake). Gilles Ritter had the same opinion:

---

<sup>22</sup> NY and Maine are in the same time zone.

Of course we know that with Geneva, we have to work in the morning. And they have to work with us [with NY] in the day-afternoon. But after that constraint, I don't see any.

**Cultural differences** are bridged:

The software team is multinational: ‘a lot of the people in the Geneva team are actually not Swiss. There are guys with Spanish origin, guys from other places. But, I would say, on a daily basis it doesn’t change the way that things are done. And I think these differences are not obstacles any more’ (Anthony Cake).

## 5.4 CONCLUSIONS

In this chapter the analysis and results of the LeCroy case study were presented and discussed. Managerial practices and quotations from interviews illustrating these managerial practices and their contribution to success, as perceived by the interviewees, were presented.

The results of the case study illustrate that interviewees considered four factors suggested in the theoretical lens, and the fifth factor (components management) that emerged from the data, as contributing to success in GD CBD.

In terms of managerial practices, *inter-site coordination* between NY and Geneva was effective and efficient: first, work was divided according to where expertise was located. Skill-based division of work was possible because team members had experience of working together from dispersed locations and had built relationships. This reduced the chances of misunderstandings and conflicts. Second, in order to increase awareness and keep remote teams updated all the time, systematic communications were organised on different levels (between managers and developers). Third, flexible project management techniques were adopted to accommodate everyday dynamics.

In relation to appropriate *tools and technologies*, first, interviewees stressed the importance of the ICT infrastructure. Second, standardization and centralisation of

software development tools enabled remote teams to work in one single development environment, and use similar tools and methods at all remote locations. Third, software development tools supported rapid update of changes by automatically (four times a day) building components that had changed; this enabled the utilisation of time-zone differences to speed the development process. Fourth, using various collaborative technologies, team members in LeCroy did not feel the differences between working with colleagues at the same office, and colleagues at a remote location.

Regarding *social ties*, in LeCroy trust and rapport between remote counterparts were developed, first, because the software team had a long history of collaborating over distance; and second, because in the beginning of the project all team members had an opportunity to meet in person in an informal environment.

Concerning knowledge sharing, in the LeCroy team transactive memory and collective knowledge were developed through the shared experience of working together over distance. LeCroy managers emphasised the importance of systematic communications and interactions (e.g. short visits and meetings in person) in order to further facilitate knowledge sharing between remote team members.

It was particularly remarkable how *components management* was organised in LeCroy: in order to maximise reuse across products, they invested time and resources in analysis to identify the most common functionalities for the product family they intended to develop. This design-for-reuse strategy enabled the LeCroy team to achieve the benefits of reuse and be more efficient in developing new products based on the Maui platform.

The LeCroy case clearly illustrates that the possibility of components reuse changes the concept of *product* in the software industry. What is a *product*? Is a component procured from a component market a *product*? Or, is a CB system that comprises commercial and in-house developed components a *product*? As Larry Salant said about products based on the CB Maui architecture:

What is the product? That, I guess, is really the key. So the products are – we have X15 as a product, WaveMaster or Aladdin is a product. But most of the components are the same in both. These are literally hundreds of these components.

This chapter presented and discussed the LeCroy case study. In the next chapter the SAP case study will be presented and discussed.

## CHAPTER 6 CASE STUDY OF SAP

*To span a project crossing Palo Alto, India and Germany is a nightmare for the people who have to work on this. 13 hours' time differences. There is no overlap. It is a pain. So, you have to have really good reasons to do something like this.*

(Stefan Mueller, Director of KM Collaboration Group, SAP Portal)

### 6.1 BACKGROUND

#### 6.1.1 BACKGROUND OF SAP GLOBAL ORGANIZATION

Founded in 1972, SAP is a recognized leader in providing ERP and other collaborative business solutions, industry-specific and cross-industry solutions, for small and medium-sized businesses, and providing technological platforms that allow for integrating heterogeneous systems. Its largest competitors are Oracle Corporation and Microsoft. SAP employs more than 32,000 people in more than 50 countries<sup>23</sup>. With operations in Bulgaria, France, India, Israel, Japan, and North America, SAP Labs integrate ideas and leading-edge technologies that address the needs of specific industries and geographic regions, and maintain SAP and its customers at the forefront of e-business success<sup>24</sup>. In 2004, revenues from software sales were 2,361 million Euros (that is 31% of the total revenue: the other 69% of the total revenue came from software maintenance, consultancy and training)<sup>25</sup>.

This case study focuses on the SAP Collaboration tools project developed by the Knowledge Management (KM) Collaboration group, which is part of the Enterprise Portal Division.

---

<sup>23</sup> From SAP web-site <http://www.sap.com/company/>; numbers are correct for April 2005

<sup>24</sup> From SAP web-site <http://www.sap.com/company/saplabs/>

<sup>25</sup> From SAP Annual Report 2004, this is the latest financial information available



### **6.1.2 BACKGROUND OF THE PROJECT AND PRODUCT UNDER STUDY**

The goal of the SAP Collaboration tools project was to develop a comprehensive collaborative platform that would enable both individuals and teams in different locations to communicate in real time and asynchronously, and to support the teamwork of any distributed project teams. The SAP Collaboration tools were developed to be part of the next generation application and integration platform (that is, SAP NetWeaver), and to allow integration with various tools of different providers.

The architecture of the SAP Collaboration tools aimed to be component-based, to allow independent upgrade of different features and, as a result, more flexibility in customizing solutions for specific customers and reduced time-to-market for new versions.

The development of SAP Collaboration tools started in September 2001. By June 2002, the first version of SAP Collaboration tools was released and the group was working on the second release.

#### ***SAP Collaboration tools***

SAP Collaboration tools provide groupware capabilities and support synchronous (real time) and asynchronous communications. Groupware capabilities include virtual work spaces (collaboration room), team folders and discussions lists, a team calendar, task assignment and tracking. They offer real-time collaboration capabilities such as desktop and application sharing that enable online meetings, remote support and co-browsing; chat; email; and video and audio conferencing capabilities, e.g. voice-over IP. Furthermore, SAP Collaboration tools offer a unified calendar function that enables task coordination (e.g. to schedule meetings) and synchronization with user's personal calendars in MS Exchange or Lotus Notes.

SAP Collaboration tools provide individuals and groups with a single point of access for documents and information sharing: 'collaboration capabilities retain

project information in context and within one location, which currently is most likely distributed among file servers, email accounts<sup>26</sup>. Information located in different places is delivered to a user on one single screen via SAP Portal<sup>27</sup>.

### ***Software components in SAP Collaboration tools***

SAP Collaboration tools have a CB architecture that is open and extensible. The components can be integrated with third-party collaboration tools like WebEx from WebEx Communications Inc., so portal users can collaborate with non-portal users. This allows users to work with familiar tools, protecting their existing technology investments.

Christoph Thommes (development architect) explained about the components included in the SAP Collaboration tools:

These are rather small components compared to something like a component in an ERP [solution] like a financials or HR as a component. We have smaller components: for example, email details are in one component, a portal component. This is a stand-alone component in the sense that it can run stand alone, and within the Portal as well. And it can be replaced by a functionally equal component, the system will still run as it did before. There are thousands of components in the product.

Information from different sources that is consolidated on the screen of the user is generated by portal components, which sit on top of the portal platform. The portal

---

<sup>26</sup> From SAP web-site

<sup>27</sup> SAP Portal ‘provides people-centric integration of all types of enterprise information, including SAP applications, third-party applications, databases, data warehouses, desktop documents, and Web content and services. It provides employees, supply chain partners, customers, and other communities with immediate, secure, and role-based access to key information and applications across the extended enterprise’ (from SAP web site). A user sees all this information on one screen.

components are called iViews<sup>28</sup>. Each element of the screen is a component. For example, Christoph Thommes explained that three views included in the Outlook mailbox - a folder list (on the left side of a screen), a list of emails (on the right side), and a detailed view of an email (on the bottom of the screen) - are components. Components are packaged together within one communication package (officially called iView Studio). The communication package consists of the different iViews and ‘connectors’ that put together different components of the package. There are two types of connectors: connectors to third party components (e.g. Microsoft Exchange or Lotus, which provide groupware functionality), and connectors to the technical (Portal) platform that provides the user interface (i.e. images that the user actually sees on the screen).

### **6.1.3 BACKGROUND OF THE SOFTWARE TEAM**

#### **6.1.3.1 WORKING EXPERIENCE IN A GLOBALLY DISTRIBUTED ENVIRONMENT**

In September 2001, when the Collaborative tools project started, key players (managers and architects) and team members from remote locations did not know each other. They did not have a history of working together. Some of the team members had previous experience of working in a globally distributed environment, but not necessarily with Indian / German / American cultures: for the majority of key players and team members this cross-cultural setting was new.

The geographical distribution of the Collaborative tools project between Germany, India and USA was the result of a merger. As Stefan Mueller (director of the KM Collaboration group) explained: ‘To span a project crossing Palo Alto, India and Germany is a nightmare for the people who have to work on this. 13 hours’ time

---

<sup>28</sup> In the SAP Glossary, iView is defined as a ‘self-contained, XML-based presentation element. A well-defined set of interfaces displays content and the personalization of the content elements presented as part of portal page’.

differences. There is no overlap. It is a pain. So, you have to have really good reasons to do something like this'. For the KM Collaboration group the key reason was a merger between SAP and Top Tier. As Stefan Mueller explained:

I didn't have an alternative: we inherited already working teams from totally different set-ups, and, based on this merger, we consolidated them at that time. [...] If I had really started out a project from scratch, I would have done it differently. But that was no question – if you are merging, you are not starting from scratch, but you have teams or locations and try to set up something that way and form a unit that is at the end of the day able to execute, somehow.

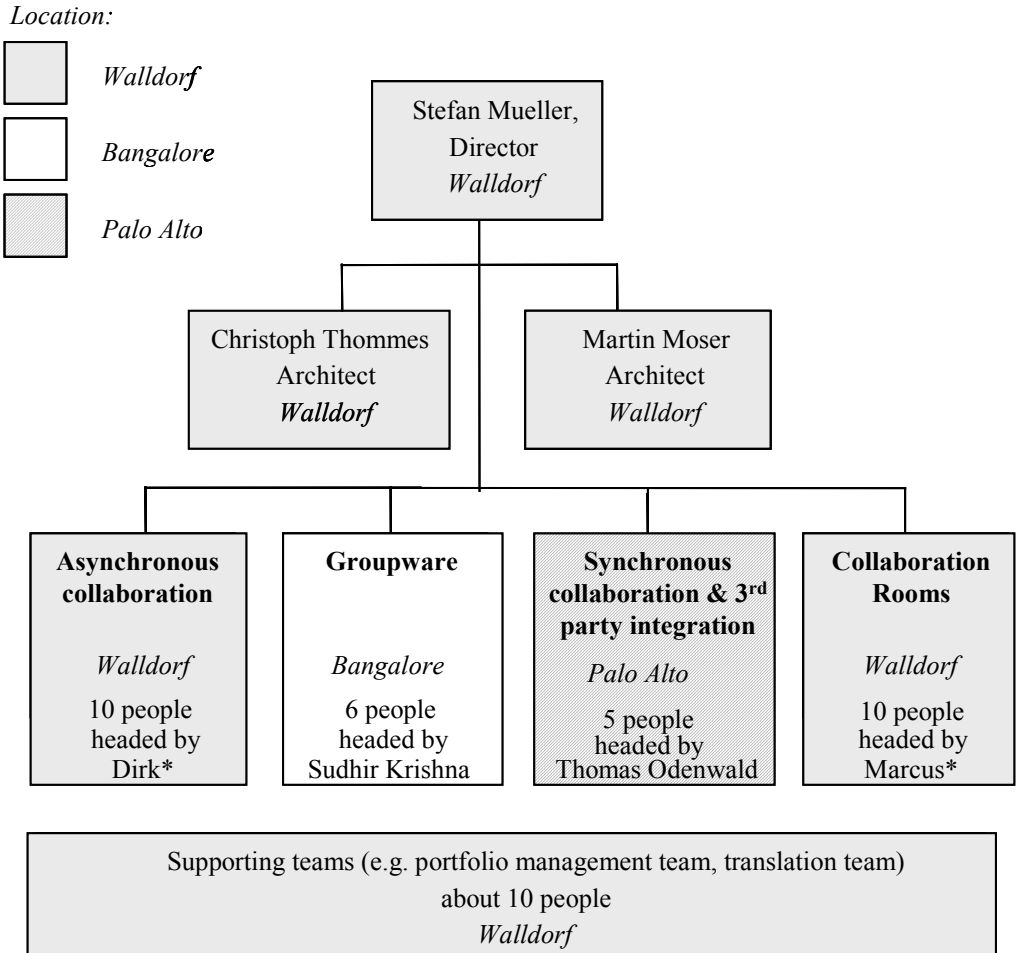
By the time the interviews in Germany were completed, in June 2002, the key players had been working together for 9 months.

#### **6.1.3.2 ORGANIZATIONAL STRUCTURE OF THE SOFTWARE TEAM**

The KM Collaboration group, where the case study was conducted, is part of SAP Portal. SAP Portal is a product organization. Different groups are responsible for different parts of a product (solution). The KM Collaboration group is responsible for SAP Collaboration tools, which are part of mySAP Enterprise Portal solution: 'we are responsible for different collaborative tools within every release. We have product cycles. Within these product cycles, we have currently several tools we have to deliver' (Stefan Mueller).

A schematic illustration of the organizational structure of the KM Collaboration group is presented in Figure 25.

**Figure 25: Organizational structure of KM Collaboration group (as of June 2002)**



\* The last names of Dirk and Marcus are not disclosed for confidentiality reasons

Stefan Mueller is director of KM Collaboration group. He is the overall project leader and responsible for delivering collaborative tools. Development managers of each team report directly to Stefan Mueller. Two development architects, Christoph Thommes and Martin Moser, work on the conceptual design of the

architecture. Their responsibility is to drive the architectural design and ensure that everything fits together.

From a geographical perspective, the software team is distributed between three locations (numbers are correct as of June 2002), and each team is working on a different part of the Collaboration project:

- 1) Walldorf (Germany): head office with 2 teams that work on asynchronous collaboration and SAP Collaboration Rooms: 10 people each
- 2) Bangalore (India): develops Groupware: 6 people
- 3) Palo Alto (USA): develops synchronous collaboration and third party integration: 5 people

Furthermore there are various supporting teams, like the portfolio management team and the translation team, which include in total about 10 people. These teams provide partial support for direct development. They are most of the time assigned to one specific product, but they are separate branches of the organizational chart and so report to different managers.

## **6.2 DATA COLLECTED**

Data was collected from a variety of sources: (i) interviews; (ii) internal project and company documents, and external reports and press releases; (iii) direct observations in the Bangalore and Walldorf offices, i.e. ten days in the Bangalore office (February 2002), and four days in the Walldorf office (June 2002); and (iv) informal conversations with managers and software engineers. Table 7 summarizes the names of interviewees, their roles, location (Bangalore or Walldorf team), and details of interviews and other communications for data collection purposes (roles are correct for June 2002).

**Table 7: SAP: Interview and data collection details**

<b>Name</b>	<b>Role</b>	<b>Location</b>	<b>Interviews and other communications for data collection purposes</b>
Alain Lesaffre	Head of corporate Research of SAP and quality manager	Bangalore	<ul style="list-style-type: none"> <li>• Interviews in February 2002 in Bangalore office, and in June 2002 in Walldorf office</li> <li>• Review and comments on the report about the team-building event, and draft of a case study</li> <li>• Follow-up for feedback and additional information</li> </ul>
Stefan Mueller	Director of KM Collaboration group	Walldorf	<ul style="list-style-type: none"> <li>• Interview on 4/6/2002 in Walldorf office</li> <li>• Follow-up by email with review of team-building exercise report</li> </ul>
Sudhir Krishna	Development manager of the Bangalore team	Bangalore	<ul style="list-style-type: none"> <li>• Interview on 15-28/2/2002 in Bangalore office</li> </ul>
Christoph Thommes	Development architect, contact person for Bangalore team	Walldorf	<ul style="list-style-type: none"> <li>• Interview on 4/6/2002 in Walldorf office</li> <li>• Review and comments on the report about team-building event, and draft of a case study</li> <li>• Follow-up by email with clarifications and additional information</li> </ul>
Akhilesh Mahto	Developer	Bangalore	<ul style="list-style-type: none"> <li>• Interview on 15-28/2/2002 in Bangalore office</li> </ul>
Jyothi Kumar	Senior developer	Bangalore	<ul style="list-style-type: none"> <li>• Interview on 4/6/2002 during his visit to Walldorf</li> </ul>

Empirical investigation included several rounds of data collection, which involved visits to SAP offices in Bangalore and Walldorf, and feedback sessions by phone

and email. Data collection covered a period of five months, from February 2002 until June 2002, and consisted of the following stages:

- First, initial contact and arrangements for data collection were done with the help of Alain Lesaffre, the head of SAP Corporate Research.
- Then, in February 2002 I visited the SAP office in Bangalore, where a first round of data collection took place: I conducted interviews and collected relevant documents. The interviews reflect the period from September 2001 (when the project started) until February 2002. Furthermore, interviews addressed plans and expectations for some activities in the forthcoming few months (e.g. a team-building exercise). During a visit to the SAP office in Bangalore I stayed at a SAP guest house and had an opportunity to talk to German engineers visiting the SAP office, observe SAP company culture, and talk informally about issues related to the management of globally distributed development at SAP.
- Next, in June 2002 a second round of data collection took place. I visited the SAP office in Walldorf and conducted interviews. By June 2002 the first version of SAP Collaboration tools was successfully released and the group was working on the second release.
- Afterwards, in August 2002, on a request of Alain Lesaffre, I prepared a report on the team-building exercise, based on the interviews that were held before and after the team-building exercise (this exercise took place in spring 2002). The report was created as an independent (objective) reflection of my research on the topic, to be used for internal SAP purposes. The interviewees reviewed the report. Several rounds of emails with comments were exchanged with Christoph Thommes during the editing process.
- Finally, in July 2004, the interviewees reviewed a draft of this case study; their feedback was incorporated.



## 6.3 HOW SAP ORGANISES AND MANAGES GD CBD: ANALYSIS AND RESULTS

In this section the analysis and results of the SAP case study are presented and discussed. First, managerial practices perceived as important in GD CBD are presented (Section 6.3.1). Then, the contribution of these practices to success in GD CBD are assessed and illustrated by empirical evidence from interviews (Sections 6.3.2-6.3.4).

### 6.3.1 SAP CONCEPT MAP

Data collected in SAP was analysed following the approach described in Chapter 4 (Section 4.5). In the SAP case in total 16 managerial practices were perceived by interviewees as important for success in GD CBD. During data analysis these practices were classified into groups that focus on different aspects of the management of GD CBD, in accordance with four factors suggested in the theoretical lens (Figure 15). Furthermore, one more factor emerged from the data, which is *Components management*.

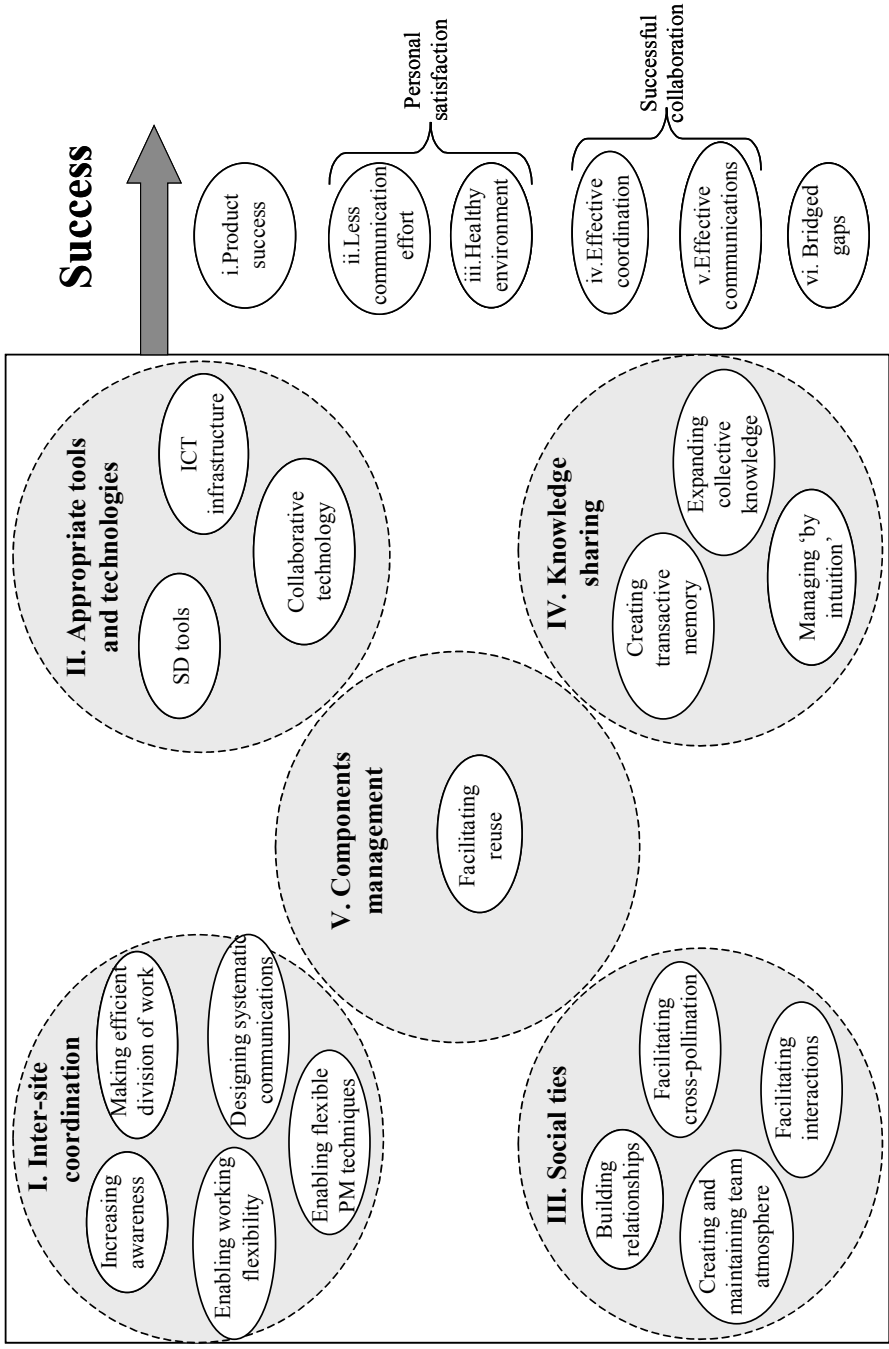
The managerial practices are classified into five groups according to the five factors and presented in the form of the concept map in Figure 26<sup>29</sup>.

In addition to the managerial practices, the SAP concept map contains categories representing evidence of success, which are (i) *product success*; then, personal satisfaction represented by two sub-categories (ii) *less communication effort* and (iii) *healthy environment*; successful collaboration represented by two sub-categories (iv) *effective coordination* and (v) *effective communications*; and the last category is (vi) *bridged gaps*. These categories and sub-categories are similar to those identified in the LeCroy case.

---

<sup>29</sup> The concept mapping approach is explained in Chapter 4, section 4.5.1

**Figure 26: How SAP organises and manages GD CBD to be successful**



In the following sections the contribution of the five factors and corresponding managerial practices to success in GD CBD in SAP will be illustrated and discussed using three types of data presentation (see ‘Within-case display’ in Section 4.5.1 and in Section 4.7).

### **6.3.2 FACTORS AND MANAGERIAL PRACTICES THAT CONTRIBUTE TO SUCCESS: CAUSAL RELATIONSHIPS**

The frequency of instances in which managerial practices were linked by interviewees with success is presented in Table 8<sup>30</sup>. As explained in Section 4.5.1, the numbers presented in the table give some indication of the strength of each relationship: the higher the number the stronger the relationship is.

---

<sup>30</sup> Table 8 is a *conceptually clustered matrix*, explained in Chapter 4, section 4.5.1: it shows the contribution of managerial practices and potential success factors to categories of success.

**Table 8: Contribution of managerial practices to success at SAP**

Managerial practices	Product success	Personal satisfaction		Success. collabor		Bridged gaps
		Less communication effort	Healthy environment	Effective coordinat.	Effective communic.	
I) Inter-site coordination	=>	=>	=>		=>	=>
1 Increasing awareness			3			
2 Making efficient division of work	1					
3 Enabling working flexibility						
4 Enabling flexible PM techniques			1			
5 Designing systematic communications		3	2		1	1
II) Appropriate tools and technologies		=>	=>		=>	=>
6 Software Development tools						
7 ICT infrastructure						
8 Collaborative technology		2	1		1	1
III) Social ties		=>	=>	=>	=>	=>
9 Building relationships		5	1		3	4
10 Creating and maintaining team atmosphere			1			
11 Facilitating interactions		1	2	1	1	5
12 Facilitating cross-pollination						1
IV) Knowledge sharing		=>	=>		=>	=>
13 Creating transactive memory among team members		2	1		1	1
14 Expanding collective knowledge of dispersed team		2	1		1	
15 Managing 'by intuition'		1				
V) Components management				=>		
16 Facilitating reuse				1		

From Table 8 it follows that managerial practices that were most often explicitly connected to success by interviewees are *building relationships, facilitating interactions, systematic communications*; and knowledge sharing practices, such as *creating transactive memory* and *expanding collective knowledge*: these contributed to the majority of categories of success.

### **6.3.3 MANAGERIAL PRACTICES: DESCRIPTION AND EVIDENCE**

In this section managerial practices perceived by interviewees as important for success in GD CBD are described and illustrated using quotations made by the interviewees (detailed description of all managerial practices is included in the Glossary of Managerial Practices in Appendix 4).

#### **(I) Inter-site Coordination in GD CBD: managerial practices**

Following are managerial practices dealing with inter-site coordination in GD CBD.

##### **1. Increasing awareness**

The interviewees indicated that increasing awareness of (i) what is going on in the company and the project and (ii) about remote team members and the environment, are important for success.

(i) Increasing awareness of the management team and key players about the ‘entire vision’ is specifically important when setting up a new organization (as in this case, when three teams were merged into one group): ‘develop the entire vision and share within the management team. Try to get all key members (managers and architects) on board’ (Stefan Mueller).

(ii) Increasing awareness concerning remote team members and the local environment is important, in particular because the teams in India, USA and Germany had not worked together before, and many of the team members were not familiar with the culture of their counterparts (for example, Stefan Mueller and

Christoph Thommes had not worked with India before, and many of the developers from the Bangalore team had not worked with Germany before). Sudhir Krishna gave a perspective of the team in Bangalore about the importance of building awareness:

For us it's more of building awareness of the whole team through Stefan, because he heads all the teams [the entire group] so he [Stefan] needs to have a good picture of how the team composition is, what each individual is like or what different people are like.

In the early stages of the project Sudhir Krishna organised a visit to Bangalore for key players from Germany and Palo Alto (Stefan Mueller, Thomas Odenwald and Christoph Thommes), who participated in the team-building exercise together with the local team. Sudhir Krishna had the following expectations for this visit:

For them [Christoph, Thomas and Stefan] it is also getting to know the infrastructure itself and the environment in which we work, because in a situation when there is a problem, then it's easy to visualise what is happening. Then, even if videoconferencing stops working all of a sudden, then you can still imagine where the people are sitting, what it looks like, you know what is going on.

## **2. Making efficient division of work**

An efficient division of work and responsibilities is considered important for success. Principles that software managers follow involve (i) giving full ownership of a product feature for each remote team and (ii) division of technical and 'social' responsibilities, which include establishing reporting channels across the globe:

(i) Work is divided feature-wise, providing full ownership and responsibility for distributed teams: 'you are responsible for what you have taken up and nobody is going to hold back anything' (Stefan Mueller). Each of the four teams has full responsibility for an entire block of functionality: groupware, asynchronous collaboration, synchronous collaboration, and third-party integration. It is

important, in particular for offshore teams, to have full ownerships of work. It gives them a feeling of being valuable and the motivation to collaborate.

(ii) A clear division of technical and ‘social’ supervision (i.e. management of local teams) between the technical architect located in Walldorf and the local development manager aims to ensure the quality of the product and effective team management. For example, Christoph Thommes and Martin Moser (two development architects located in Walldorf: see Figure 25) serve as technical contact persons for the remote teams: Christoph is a contact person for the Bangalore team, and Martin is a contact person for the Palo Alto team. The architects provide technical supervision for the assigned remote team, and are responsible for technical issues and the quality of software developed by this team. Christoph Thommes explained that because a technical architect drives the overall product architecture, he is the most appropriate person to provide technical supervision and to control the quality of the product:

I’m in a position where I have to supervise sometimes what they [Bangalore team] are doing from a technical point of view, I need to point out certain weaknesses in whatever they’re doing, from a code perspective for example (Christoph Thommes).

The local development manager of each team is responsible for team management: he divides specific assignments (tasks) between team members and resolves social issues. The development manager and team members are of the same culture. This makes it easier for the development manager to understand and deal with the team members. As Christoph Thommes explained:

I’m not responsible for people, people management is completely up to Sudhir. He deals with the team: assign tasks to the team members, reviews tasks, gives a performance feedback. If there is a technical conflict, there is an agreement that because of my role as a development architect, I’m the one to take the last final decision call on technical issues.

### 3. Enable working flexibility

The interviewees suggested that working flexibility, in terms of (i) providing flexible working conditions such as working from home and (ii) flexible working hours, are important for success. For example, Christoph Thommes explained how he uses flexible working hours to increase overlap in working hours with India:

I start quite early in the morning: they [in Bangalore] come to the office maybe at 9 something and I start at 7 something so it's 1½ hours where they cannot reach me and they stay quite long.

### 4. Enabling flexible Project Management (PM) techniques

Flexible PM techniques help to accommodate everyday dynamics. They include:

- On a macro level: Planning of major project activities (milestones)

‘We have project phases - three to six to, maybe, 12 months, depending on the part of the project’ (Christoph Thommes). On a macro level planning includes setting up clear objectives for teams (what features each team should deliver) by the project manager (Stefan Mueller) and development architects. Then, within each team planning of work is done by the local development manager:

I set up clear objectives, but then I give them [local development managers] an entire area for which they are responsible: this means that I am also not buying excuses if they don't deliver. I give them an entire block of functionality, and I give them full responsibility to plan properly, to execute, to tell them what they are actually getting developed. So I put the requirements, what we need, feature-wise, and then judge them from what they really deliver. What they do with the team, how they do it – I don't care. I tell them to send me updates, and we can adjust the plan if they want. If they don't tell me anything – I take this for granted – this is our culture, this is how we decide what we will deliver (Stefan Mueller).

- On a micro level: Flexible and not too detailed planning



As Christoph Thommes explained:

We do not plan exactly on a daily basis: maybe it's giving an estimate of how long the task might take and assign someone, who is responsible for the specific task. So it is not in days, but it is more or less weekly milestones.

## 5. Designing systematic communications

Systematic communications considered by interviewees as contributing to success.

The design of systematic communications involves the following:

- Scheduling systematic and frequent communications, such as regular teleconferences between software managers in Walldorf, Bangalore and Palo Alto, transatlantic videoconferences with all team members every couple of months.
- Communicating directly to reach an appropriate person

For example, on the question of what is most important in a globally distributed environment, Christoph Thommes answered:

Quick and direct communications as far as possible, is the most important thing. 'Direct' means: do not communicate through other people but with the people directly. If you have one contact person who distributes all the information, you lose some amount of information, just because you do not reach the right people.

- Setting up rules of communications

Setting up rules of communications helps people to adjust to communication styles and reduces misunderstandings and confusions that typically happen as a result of a different cultural backgrounds. For example:

- Agreement was reached that the Indian team members would not take it personally when Germans are too direct, because, compared to Indians, Germans usually are very direct and 'brutally precise' in communicating what

they have in mind and, typically, this is one of the biggest challenges in German-Indian teams.

- Stefan Mueller explained about his experience with Sudhir Krishna in adjusting communication styles; how helpful it was to ensure successful communications over distance:

What I did with Sudhir in the very beginning, I told him: ‘I am explicit, I am forgiving – if you tell me in the beginning that something is going wrong. Because it is not just me having to deal with an Indian team, changing my style totally. I will try to adapt but because of time-constraints I am not going to adapt exactly to what you are expecting. Otherwise you tell me if you have a problem’. That was easy. That is what we did on the face-to-face meeting when he [Sudhir] was here in Germany. Sudhir said that this is clear, and now we can see that it worked.

Furthermore, communicating over distance, it is important to make communication ‘precise’:

Being precise means being very explicit: making clear statements, especially when you are not meeting face-to-face. If you look at me when I have a telephone call, you would say ‘you are brutally precise’: I say [by phone] ‘yes’, ‘no’, ‘no’, ‘no’, ‘yes’ and there is no answer such as ‘maybe’: ‘maybe’ just doesn’t work (Stefan Mueller).

## **(II) Appropriate Tools and Technologies in GD CBD: managerial practices**

Managerial practices related to tools and technologies identified in SAP as important in GD CBD are as follows:

### **6. Software Development (SD) tools**

In order to support CBD in a globally distributed environment SD tools need to provide the following capabilities:

<b>Standardization of tools and methods across locations</b>	<p>Everyone in the KM Collaboration group uses the same tools and methods:</p> <p>We use all the same tools, so there is no difference. We even use the same Word templates [templates with project activities and related document], so even the specifications look more or less the same (Christoph Thommes).</p>
<b>Centralisation of tools and web access</b>	<p>Centralization of tools under a single environment accessible from all remote locations over the Web is important to make sure that everybody is working with the same, most updated versions. For example, Sudhir Krishna explained that SAP Intranet (called SAPNet) serves as a central place that has links to all updated information: all the documents are accessible from SAPNet while in practice they are located at Perforce – a VCS that is linked to SAPNet via a Web server:</p> <p>We use SAPNet for storing the information: we store mostly all the documents in Perforce, and we have a Web server that accesses the Perforce; this Web server is linked to SAPNet. So SAPNet is a central medium - when a user clicks on a link at SAPNet, it takes him to the appropriate machine and shows him the document (Sudhir Krishna).</p>

---

## 7. ICT infrastructure

---

ICT infrastructure implies high bandwidth reliable connections between all remote sites to support the following:

<b>Quick access to the network</b>	<p>Quick access to the network is required from all remote locations. Powerful servers are used to allow quick access for multiple users from remote locations.</p>
------------------------------------	---

---

<b>Quick and easy connectivity across locations</b>	Quick and easy connectivity between locations is important. For example, setting up internal phone lines across the globe (5 digits number between Bangalore and Walldorf) makes it easy to contact remote counterparts.
<b>Shared server and project repository and Web access</b>	<p>There is one server that can be accessed from remote locations, therefore sometimes team members pass bug fixes from one team to another to take advantage of time-zone differences:</p> <p>Typically that's what we do: if I get an email sometime in the afternoon or evening 'there is a problem, can you look into it?', I say 'OK, by tomorrow morning your time it will be done'. Because Walldorf is sleeping and at that time we log in and finish the issue, so that there is an advantage (Sudhir Krishna).</p> <p>There is also a central project repository on SAPNet accessible over the Web; it ensures that everyone has updated information:</p> <p>On the SAPNet we have our own let's say, branch for SAP Portal where all the things are kept up to date: all the documents, questions, answers, everything is maintained there, so we have our sort of central repository (Akhilesh Mahto).</p> <p>A project plan is also accessible from SAPNet: 'On SAPNet we have a central place where we update the project plan, where we set the deliverables' (Sudhir Krishna).</p>

## 8. Collaborative technology

The following are collaborative technologies used by LeCroy team to collaborate successfully over distance:

---

**Phone and teleconferencing**      The phone is used for urgent matters, regular updates between managers and to resolve misunderstandings. For example:

It's a lot easier to get a direct response than to just send emails back and forth saying 'here you wrote this and I interpret this as this, and my response is this', and if we talk, it's a lot easier to communicate (Christoph Thommes).

---

**Application Sharing**      Typically an AST is used remotely (i) for discussions that involve showing slides (usually, in such situations remote counterparts use AST to show presentation slides, and at the same time they use the phone to explain the slides and to discuss issues and questions); and (ii) for discussing technical issues (e.g. code reviews, debugging): in this case the AST is used for taking control of a computer remotely.

---

**Videoconference**      VC sessions that involve managers and developers in all three locations are used to discuss progress and other issues; they are organised twice a month. For example:

Whenever a new colleague joins in our team or any of the teams in the other locations, in the next VC which we have, we have an introduction round like 'these are new colleagues that have joined'. So though you have not met them physically, you get to know that this is the person, he exists there, things like that (Akhilesh Mahto).

For design reviews, sometimes, a VC is organised:

If no major changes required, then it is not necessary to have a VC. But if the issues are critical, then we certainly need to have a VC. To discuss 'why do you propose such and such a design?', it's better to talk face-to-face [over VC] and explain face-to-face, than to keep sending emails (Akhilesh Mahto).

---

<b>Email</b>	Email supports low priority tasks and issues, and tasks that cannot be completed in real-time because of time-zone differences. Email is used as documentation (record) as well: for example, as Christoph Thommes explained:  I prefer email, because I'm not too good at making notes during conference calls, so for me it's easier to just write it in an email and have it in my Sent items, which I never delete. So when a question arises after maybe a month or maybe half a year, I can still look into my emails and I can quickly search my emails for specific topics.
<b>Intranet</b>	The team has access to SAP internal Intranet environment (SAPNet), where internal documents and other relevant information are posted.

### **(III) Social Ties in GD CBD: managerial practices**

According to the opinions of interviewees, rapport and trust contribute to success, as illustrated by the following quotations:

#### ***Contribution of Trust***

Right now I know people pretty well. With India I had a problem in the beginning, until Sudhir and I got to the level of confidence that he is able to interpret my reaction and I am able to deal with him (Stefan Mueller).

#### ***Contribution of Rapport***

I need to have good relationships with the people I am working with [...] the better you know the people the easier it gets. I know Sudhir and Thomas I think by now quite well (Christoph Thommes).

Following are managerial practices that focus on social aspects and facilitate rapport and trust between remote counterparts.

## 9. Building relationships

Interviewees consider building relationships between remote team members very important for success.

For example, it was important to build relationships between the team in Bangalore and Christoph Thommes (located in Walldorf), who works closely with that team. Sudhir Krishna (development manager of the team in Bangalore) explained that it was very important that Christoph met and got to know personally the whole team in Bangalore, and team members got to know Christoph, because:

Christoph is the person to whom all of us email, regarding any technical issue. We don't say 'you all have to mail me then I will mail to Christoph', we all email to Christoph directly (Sudhir Krishna).

## 10. Creating and maintaining team atmosphere

Creating and maintaining a team atmosphere is important, in particular for offshore teams in Bangalore and Palo Alto.

Stefan Mueller explained that it is important to show team members that:

there is no fear, that I am not playing tricks with them, that I am trying to be an ambassador, that we have visibility, that our product is wanted, that we get the respect of the other teams, that we are properly embedded within the overall management group, that there is enough room to grow – this is what they [team members] expect [from the head of the group]. It was pretty hard to establish among them [all teams] a 'no-fear environment' because they see me at videoconferences and that's like a lecture, this is the only way to do a videoconference with about 30 people at 3 locations: not much discussions, or the communication just fails.

Visits to remote locations help to create and maintain the team atmosphere. For example, during a team-building exercise, letting team members of the Bangalore team meet and spend some time together with key people, specifically with the head of the group, Stefan Mueller, gave the team members a feeling of belonging,

of being part of the KM Collaboration group, and equally important, as the other teams in Palo Alto and Walldorf. This was one of the goals Stefan Mueller had during his visit to India - to give confidence to the team members in Bangalore that they are important and they are part of the KM Collaboration group and part of SAP Portal: ‘the team-building for me was for them to show ‘yes, you as a remote location are valuable’, to give the overall organisational confidence’ (Stefan Mueller).

## **11. Facilitating interactions**

Facilitating interactions between people at remote locations is important. It includes (i) facilitating personal face-to-face interactions and (ii) organising regular and frequent interactions over distance.

For example, personal face-to-face interactions are particularly important in the beginning of a new collaboration, as in this case when several teams were merged into one group:

With Sudhir we are now about 9 months working with each other. With Thomas in the USA, in Palo Alto, it’s the same. With Marcus, in here [in Walldorf], it is much easier to get accustomed to working habits, because he is just sitting in the next office right now. And then it is much easier for him to understand how the director [Stefan Mueller] reacts, and why he reacts, why he is so pushy or not pushy, or doesn’t react. That what I learned with Sudhir. But just because Sudhir came over here. So, this ‘develop confidence’ is something you have to set up once in the face-to-face meeting, or even a longer stay (Stefan Mueller).

## **12. Facilitating cross-pollination**

Interviewees considered cross-pollination (i.e. that people from the one group spend significant amounts of time in the remote group and visa versa) to be important for success. In particular, it was helpful to deal with cultural differences



between German and Indian cultures, because, usually, it takes a long time to get to know and get used to these differences. Sudhir Krishna worked several years in Germany and knew about German culture and the German way of working and communicating before he got involved in the development of SAP Collaboration tools. Stefan Mueller explained:

Sudhir had an advantage – he was here [in Germany] for 2 years, so he already knew Germans, and this is a big advantage. The most he has to deal with is - German habits, dictatorship German habits. And that is what I knew. And that is what he also told me.

#### **(IV) Knowledge Sharing in GD CBD: managerial practices**

Interviewees consider knowledge sharing as contributing to success; in particular, building up collective knowledge through shared experiences, and creating transactive memory among team members at dispersed locations.

The globally distributed teams in Walldorf, Bangalore and Palo Alto did not have a history of working together before they were merged into KM Collaboration group. The transactive memory and collective knowledge in this group had been developing since the project started (i.e. since the merger).

Following are managerial practices seen as important for knowledge sharing between members of remote teams.

#### **13. Creating transactive memory among dispersed team members**

Creating transactive memory among team members in Walldorf, Palo Alto and Bangalore was considered important for success.

Transactive memory is important because it influences the amount of information that needs to be shared, and has an impact on the efficiency of communications, as illustrated by the following quotes:

A simple one-line question can result in a 10-page answer. It can be a very lengthy answer, or he [the person who answers] can simply cut it

up by giving a one-line reply. And as to what detail you get in an answer depends on how well you know that person. Because if the person knows me very well and he knows in what areas I am working, then he can decide how much information I will need. Is one-line good enough for him or should I explain to him over three pages so that he knows what is happening? (Sudhir Krishna).

Furthermore, in a globally distributed team transactive memory enables staff to coordinate efficiently across locations. For example, Christoph Thommes explained:

What I did in the past was - this was in the very early phase of the project, I sent requests only to Sudhir and he would distribute the issues between people. But by now, after 6 months, I know quite well what everybody is doing. So after a time, you just know who's doing what.

#### **14. Expanding collective knowledge of the dispersed team**

Expanding collective knowledge of the dispersed team is important for success; in particular it was important to create collective knowledge about differences in the national cultures of people involved in the project: Indian, German and American cultures. For example, during team meetings people are encouraged to reflect on their perception of cultural differences they experienced when visiting a remote location and/or communicating with remote counterparts, as Jyothi Kumar experienced during his visit to Walldorf.

Furthermore, in the beginning of the project, there was a knowledge / experience gap between people involved in the project:

People have different profiles: here [in Bangalore], the maximum experience is 5 years. But if you take these three colleagues travelling to the team-building exercise [Stefan Mueller, Christoph Thommes and Thomas Odenwald], the two of them have about 12-15 years of experience, and the minimum experience here [in Bangalore] is about 2½ years, so that's a huge experience gap that they have to bridge (Sudhir Krishna).

One of the goals of the team-building exercise organised in the beginning of the project was to bridge this gap and create collective knowledge in the globally distributed team. As Stefan Mueller reflected:

It [team-building] was a pretty good experience for myself: learning the culture and also how the team internally works. So my understanding of what you can expect from the team, and what you cannot expect, is very important for the project.

### 15. Managing ‘by intuition’

Management ‘by intuition’ is based on catching signals and sensing (feeling) that something is working or not working properly. The ability to manage ‘by intuition’ is important for success. It is illustrated by the following extract from the interview with Stefan Mueller, who had nine years of experience in the management of software development at the time the interview was conducted:

Stefan Mueller (SM): Quality, time-line, this is what you see. This way you feel if something is not working properly.

My question (JK): Could you also feel if something is not working properly in the very beginning?

SM: No. If I had led this unit 5 years ago, I would be in deep shit. This is what you have to learn. With experience you know the signals: they are not written, they are not formal and nobody tells you: ‘it is something missing today, it is too quiet, it cannot be that quiet because there have to be some problems’. And it is experience, it is guidance, connecting with other people, supporting, helping them to overcome these problems. But for the other areas that are working – don’t touch them.

JK: To sense this, you probably need to know very well your development managers and architects?

SM: Sure, you also need to know what they tell you, what they don’t tell you, how they react. You read between the lines.

To enable management ‘by intuition’ in globally distributed environment, a manager needs to know his/her subordinates personally and to have a rapport with

them. Then he/she might be able to catch and interpret signals when a subordinate sends too many or too few progress reports, or perhaps too many or too few clarification requests.

### **(V) Components Management in GD CBD: managerial practices**

In addition to the four factors suggested in the theoretical lens, *components management* emerged from the SAP data as factor contributing to success. Following is a managerial practice seen as important to enable reuse of components.

### **16. Facilitating reuse**

It was indicated that facilitating the reuse of *knowledge* and reuse of *components* across locations are important for success in GD CBD. For example, as Akhilesh Mahto explained:

The team in Walldorf should be aware of what is being developed in Bangalore or Palo Alto, so that we don't reinvent the wheel again and again. So we basically communicate about what are the things that are being done, and is there something reusable which we are developing, or have they developed something which somebody else can use. Maybe some of the packages which we have developed might be useful for the team in Palo Alto. Maybe they are developing some application which needs a package smaller, maybe a half a package can fit into that. Then you are not rewriting the whole product again and again. Maybe they can just use our package available, make some changes according to what they need, and use it. For things like that we need to interact with each other.

### **6.3.4 SUCCESS IN GD CBD: EVIDENCE**

This section presents evidence collected in interviews about success achieved in the studied case. The evidence (quotations from interviews) is presented according to the categories of success illustrated in the SAP Concept Map (Figure 26).

## **i. Product Success**

We just went through a merger, so setting up a global project was not an easy task. Despite all the difficulties we managed to have a successful second software release in 8 months (Stefan Mueller).

Furthermore, there is external evidence of project success:

- According to JupiterResearch, a leading research and consulting company in emerging technologies, SAP Enterprise Portal is the third largest software solution, with 17% of the USA market in 2002. The studied project developed SAP Collaboration tools as one of the main features of the SAP Enterprise Portal.

## **Personal Satisfaction**

### **ii. Healthy environment**

The team-building exercise was a way to show that we care about remote locations. The end result of that exercise was that the entire team [globally distributed] feels more comfortable to work together. Now they know each other and trust each other better (Stefan Mueller).

### **iii. Less communication effort**

Jyothi Kumar (senior developer from Bangalore team) expressed his team's perspective on the team-building exercise:

The team-building exercise improved relationships among the KM Collaboration group [between the team located in Bangalore and their remote colleagues], because earlier communications were only in a formal way, and after the team-building activity we really knew people much better, it became easier to communicate and communications became more informal.

As Christoph Thommes explained:

It's a lot easier to pick up the phone, from my experience, to pick up the phone and call someone if you at least met him once. Or if this is not possible due to cost reasons, at least see him via the

Videoconference. If you see someone, at least for me it's completely different to communicate later via the phone.

## **Successful Collaboration**

### **iv. Effective coordination**

I am not controlling in details. On the other hand, I am pretty much in line with their daily activities, and take action if I see problems popping up (Stefan Mueller).

### **v. Effective communications**

After the key players visited the Bangalore site and got to know remote team members personally, centralized communications (via Sudhir Krishna) were replaced by direct communications. Christoph Thommes explained:

From a code perspective for example, what I did before I met all of them [team in Bangalore] in person was to send all things to Sudir and he was the one to distribute it within the team, and this has changed now. I address most of the things directly to the team members.

## **vi. Bridged geographical, time-zone and cultural gaps**

### ***Geographical distance***

Geographical distance creates limitations for face-to-face meetings. The costs of travel limit opportunities for team members to meet in person. To overcome geographical distance the following practices are adopted:

- For managers: 'we generally keep travelling at least once in every three months. But if there is a need or there is an urgency, then we travel any time' (Sudhir Krishna).
- For developers: 'the idea is that every developer travels across [to Walldorf] and meets everybody once for the reason to get to know each other in person rather than just by name' (Sudhir Krishna).

***Time differences*** cause problems, but sometimes can be used as an advantage:

Time differences between Germany and India are not seen as a problem, and are sometimes even used as an advantage:

Sometimes we find it advantageous, especially if there are demo systems in Walldorf, if you are to send data for demos, its really easy for us. Because by that time Walldorf is sleeping, we have 4½ hours where we can finish our stuff and log off. And then people in Germany fight for that (Sudhir Krishna).

Also problems (bug fixes) sometimes are passed across time-zones.

However, people in Waldorf and Bangalore find it very difficult to work with Palo Alto:

- ‘for us it is definitely a problem: we sleep and they wake up’ (Sudhir Krishna)
- ‘because you cannot communicate the information in time’ (Christoph Thommes).

For the team in Palo Alto, which mostly has to communicate via email, answers are always delayed (at least for one day): they do not have a possibility to call when a question arises, but need to plan calls in advance:

The biggest disadvantage is that both Walldorf and India are sleeping when they are awake, so the information flow for them is even more difficult - they have to specifically request for a telephone call and they have to plan it in advance (Sudhir Krishna).

One of the ways adopted to reduce time-zone differences was to fly some people from India to Walldorf during the last stages of the project (before product release) so that they could finish the project working from only two locations: Walldorf and Palo Alto. This reduces time differences from 13 hours to 9 (that is the time difference between Walldorf and Palo Alto). As Christoph Thommes explained:

Jyothi is here at the moment, he will stay another 4 weeks, and Sudhir and Akhilesh will arrive in two weeks [to Walldorf]. They are going to finish a project here and work closely together, which is a lot easier. Since the time difference of 11 or 13 hours, depending on when you

start coming to the office, it's easier to finish the project here [in Walldorf].

**Cultural differences** are bridged to a great extent:

Team members in Bangalore are Indian: team members in Walldorf are mostly German. The team in Palo Alto is multinational: 'there are Chinese guys, Ukrainian guys, somebody from India, a German manager of the team plus also other units' (Stefan Mueller). As indicated by the interviewees, differences between German and Indian cultures are mostly in the way of working, way of communicating, and values. A team-building exercise helped to bridge cultural differences, as Sudhir Krishna explained:

From my perspective it was a new thing for Stefan and Christoph to get to know how Indians work, values - a cultural thing. Thomas has worked for more than two years with Indians, so he was aware of our working style.

## 6.4 CONCLUSIONS

In this chapter the analysis and results of the SAP case study were presented and discussed. Managerial practices and quotations from interviews illustrating these managerial practices and their contribution to success were presented.

The results of the case study illustrate that interviewees considered four factors suggested in the theoretical lens, and the fifth factor that emerged from the data as contributing to success in GD CBD.

In terms of *inter-site coordination*, the work was divided between the teams in Bangalore, Walldorf and Palo Alto based on product features, providing full ownership and responsibility for each team. There are two reasons why SAP gave full ownership to each of the remote teams, instead of dividing the work based on the expertise of individual team members. First, because when the project started remote teams did not have knowledge about the product: collaborative tools were developed from scratch. Second, because teams had just merged into one group,



they did not have a history of working together. Thus, giving full ownership to each of the remote teams reduced dependencies and, therefore, the need for coordination between the teams.

Moreover, systematic communications between key people (architects located in the headquarters and development managers of the remote teams) were important to ensure quality of the product: that components developed by the dispersed teams fit together.

In relation to *appropriate tools and technologies*, remote teams used similar tools and methods across locations. Various collaborative technologies were available for dispersed team members, for example internal phone lines (a 5 digit number) between Bangalore and Walldorf made it easy to contact remote counterparts.

Regarding *social ties*, in SAP three teams were merged into one group in the beginning of the studied project, and members of these teams had to build trust and rapport from scratch. Team-building exercise and short visits were organised to give developers and key players an opportunity to meet in person in an informal environment and get to know each other. This helped to create transactive memory and build relationships among the team members.

Concerning *knowledge sharing*, in the beginning of the project the SAP team did not have a transactive memory and collective knowledge. Therefore, interactions were particularly important to create transactive memory and collective knowledge about the cultures of the remote counterparts and of the evolving product. Interviewees from SAP suggested that knowing who knows what at a remote location enables the organisation to reduce development lifecycle because response is quicker when team members know whom to contact for a specific problem. Moreover, the importance of intuition for managing GD CBD projects was emphasized. To be able to manage ‘by intuition’, extensive experience in the management of software development in general and globally distributed projects in particular is required.

In regard to *components management*, globally dispersed teams organised formal meetings, usually using VC tools, to discuss what each team has developed and to identify an opportunity to reuse knowledge and/or software components (applications).

This chapter presented and discussed the SAP case study. In the next chapter the TCS case study will be presented and discussed.



## CHAPTER 7 CASE STUDY OF TCS

*We all speak Quartz language. It is a loss for us if somebody leaves Quartz because for somebody new it will take time to learn Quartz.*

(Pankaj Khurana, Offshore Project Leader, TCS)

### 7.1 BACKGROUND

#### 7.1.1 BACKGROUND OF TCS GLOBAL ORGANIZATION

Tata Consultancy Services (TCS) was established in 1968 as a division of Tata Sons Ltd (Tata Group). TCS is one of the biggest Indian software companies: it specialises in IT consultancy, services, and business process outsourcing, and is recognised among the 25 top IT consultancy companies in the world. TCS employs over 28,000 people in 32 countries, and it has 26 development centres all over the world: in India (11 centres), USA (8 centres), Canada, UK, Uruguay, Hungary, Australia, China and Japan. Sixteen of these centres have been assessed as operating at Level 5 maturity on the Software Engineering Institute's Capability Maturity Model (SEI CMM) scale. The main TCS industry practices include banking, financial services and insurance, telecom, manufacturing, transportation, and retail and consumer goods. The main service practices of TCS include e-business, architecture and technology consultancy, process consultancy, and application development and maintenance. TCS reported total revenues of 5,985 crores of rupees (about \$1,368 million according to the conversion rate in April 2005) in the nine months ended December 2004 (this is the latest financial information available)<sup>31</sup>.

---

<sup>31</sup> This section is based on the TCS web-site, corporate brochure and financial reports: numbers are correct for April 2005.

### 7.1.2 BACKGROUND OF THE PROJECT AND PRODUCT UNDER STUDY

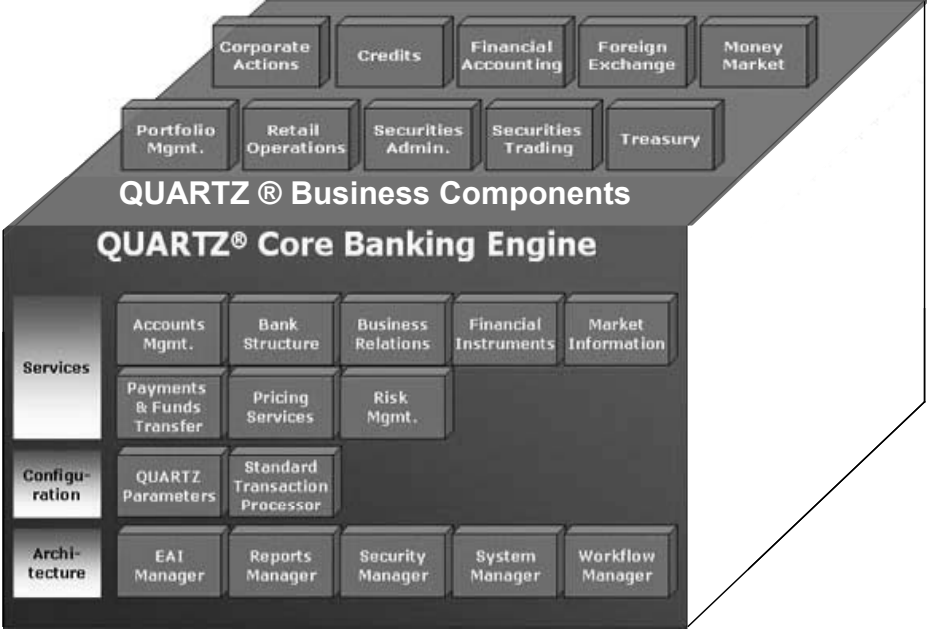
This case study concerns the development and implementation of Quartz, an integrated financial platform aimed at providing solutions for financial institutions such as traditional and internet banks, brokerage/securities houses and asset managers.

#### *What is Quartz?*

Quartz is an integrated package and banking platform for the international financial industry. It was developed jointly by TCS and TKS (Teknosoft, a Swiss-based company that specialises in financial services), through a partnership in which the technical knowledge and experience of TCS in providing computing services was combined with the business knowledge of TKS of the financial industry and banking.

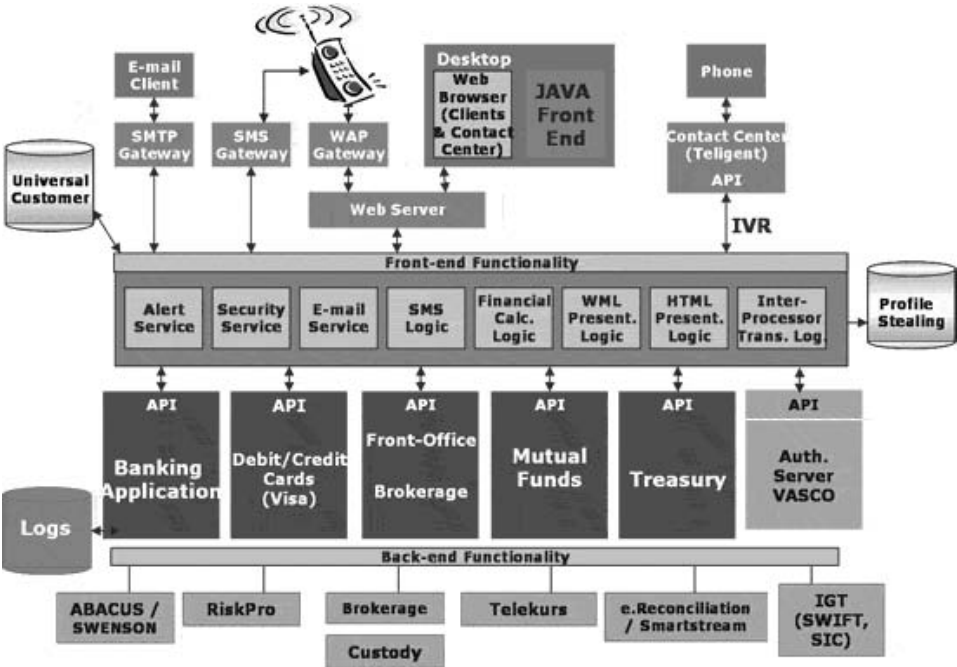
Quartz consists of a collection of architectural and business components that can be integrated with third party components to provide a solution according to the requirements of a specific customer. The Quartz architecture consists of (1) core banking components that are integrated into a Core Banking Engine, and (2) business components, which are added as an additional layer on the top of the Core Engine, as illustrated in Figure 27. *Core banking components* provide core banking functionalities, such as business relations, financial instruments, market information and parameterisation information: they can be (easily) adapted to a specific bank environment and integrated into the legal framework within which the bank operates. Additional functions offered as *business components* may be installed, omitted or replaced with third-party components. Together these two types of components ensure that the Quartz architecture provides a flexible package of the entire banking application.

**Figure 27: Quartz component-based architecture**



The first implementation of Quartz took place in 1998. In March 2002, when I visited the Gurgaon office of TCS, TCS was implementing Quartz in several organizations. Two implementations, at Skandia Bank Switzerland (SBS) in Zurich, Switzerland, and Dresdner Bank in San Francisco, USA, were approaching completion: both projects were at the stage of end-user testing. A few more Quartz implementations were in the very early stages. By April 2005 more than 40 installations or implementations of Quartz were in progress. An example of Quartz implementation (i.e. customization of Quartz for a specific customer) is shown in Figure 28.

Figure 28: Technical overview of Quartz implementation (modified example from TKS web site <http://www.tks-teknosoft.com/implementation.html>)

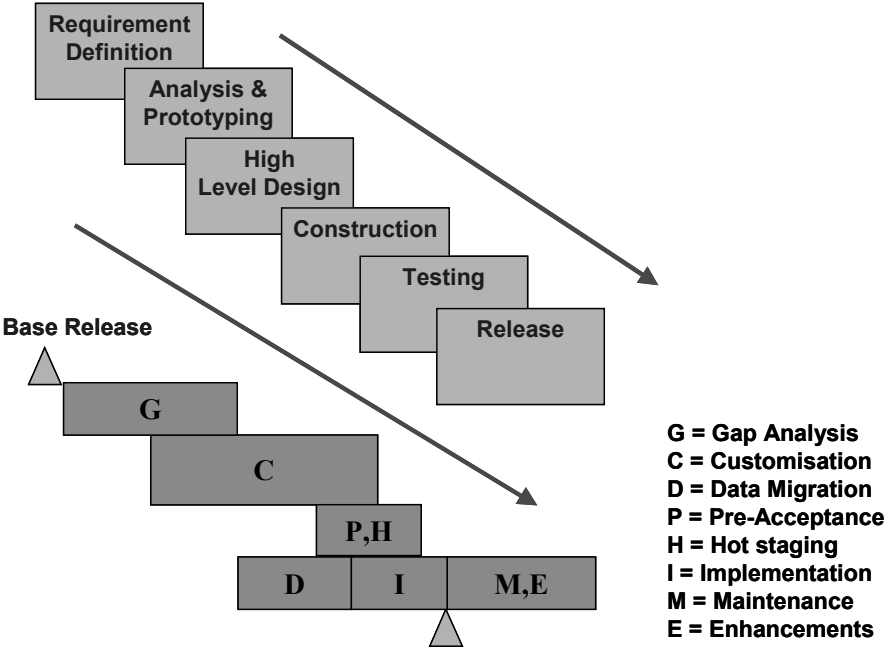


**Functionality supported:**

- Cash balance forecasting
- Realized P&L calculations
- Integrated Securities broking and trading
- Asset Allocation
- Stock-watch alerts and standing instructions
- Position transfers and electronic payments
- Online Corporate Actions
- Automated e-mail contract and statement delivery
- Real-time update of settings
- Mobile Access
- Call Center, IVR Support

The typical methodology adopted by TCS for Quartz product development and solution implementation is illustrated in Figure 29.

**Figure 29: Quartz: product development and solution implementation methodology (adopted from TCS internal documentation)**



In this case study two Quartz implementation projects are investigated: (1) implementation in Skandia bank in Zurich and (2) in Dresdner bank in San Francisco. Both projects are concerned with the implementation of Quartz, therefore they are analysed together as one ‘embedded’ case study (Yin 1994): in this case the two projects are sub-units of analysis (as explained in Chapter 4, Section 4.2).

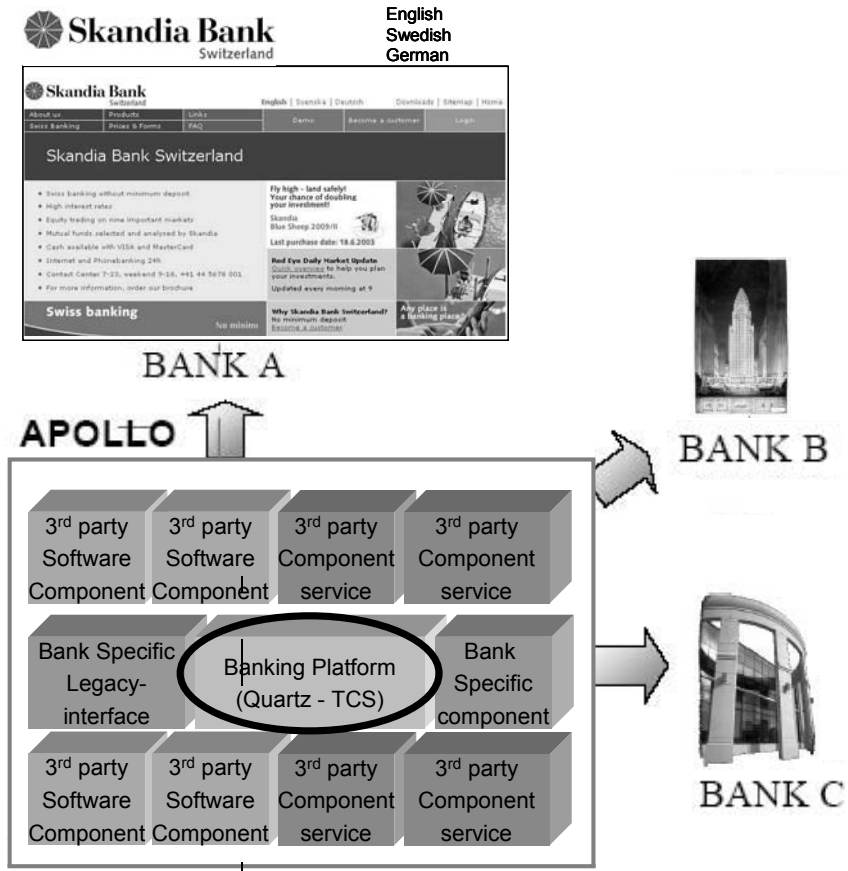
***Project 1: Skandia***

The Skandia project started in October 2000. It involved the development of Apollo, an Internet-based banking platform, and the implementation of this platform in Skandia bank. The Skandia project described in this thesis is the first implementation of a bigger project (described by Alexandersen et al., 2003) in which Skandia Group, the customer of the Skandia project, aimed to create a so-



called ‘bank-in-the-box’ they could sell in the future to different banks<sup>32</sup>. For the first implementation of this bigger project Skandia Group had chosen its own bank, Skandia bank. Figure 30 illustrates the ‘bank-in-the-box’ Apollo platform and its implementation in different banks.

**Figure 30: Skandia project: Apollo CB architecture (adopted from Alexandersen et al. 2003)**



<sup>32</sup> Alexandersen et. al. (2003) described the Skandia project from the strategic perspective of the Skandia Group - a client of TCS. In this case study the Skandia project is described from a TCS perspective, focusing on the development and implementation of the Quartz platform in a globally distributed environment.

As shown in Figure 30, at the heart of the Apollo platform is the Quartz banking platform. The original Quartz served as a back-office: it was customised and extended to suit the needs of Skandia bank. One of the major extensions was the front-end design that involved design of the users' interface (i.e. what a user sees on a screen when he/she logs into the internet-bank, e.g. a menu with different options of what he/she can do on the screen). The content of the front-end was designed by Mogul, a Swedish company hired by Skandia. The actual programming to implement the content of the front-end was done by a Front-End group of TCS, located in Bombay.

In addition to the Quartz implementation, TCS was responsible for establishing a data and recovery centre, a physical centre to support business operations of an Internet bank: this included the management of vendors delivering third-party components, such as operating systems, hardware and service providers<sup>33</sup>. In total, more than 25 vendors located in many different countries were involved in the project, among them Salomon Smith (broker, UK), Reuters (real-time rate provider, UK), Oracle (database, USA), Sun (servers, USA) and CISCO (networks, USA).

In terms of global distribution, the Skandia project involved three main geographical locations: two offshore TCS teams in Gurgaon and Bombay, and an onsite team at the customer location in Zurich where the physical data and recovery centre had to be established. Furthermore, vendors of third party components were located in different countries.

At the time of data collection, the offshore Quartz team in Gurgaon involved six people, the offshore Front-End team in Bombay had five people, and the onsite

---

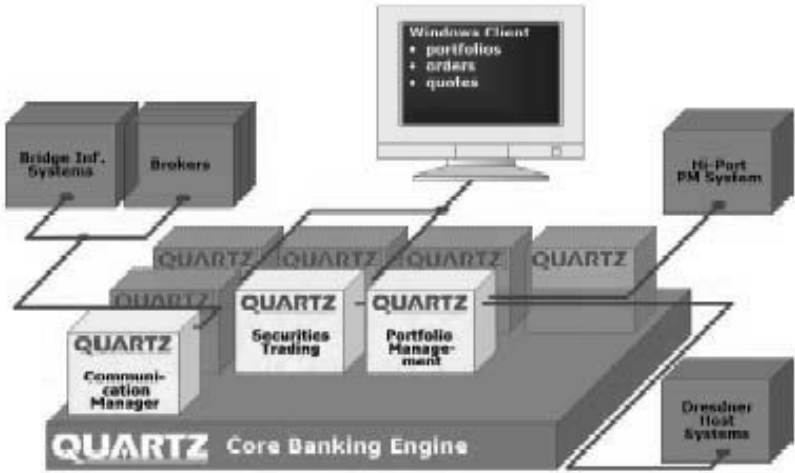
<sup>33</sup> For Skandia Group, TCS was a vendor providing the major component (i.e. Quartz as a banking platform); furthermore, it was managing all other vendors delivering components that needed to be integrated in the Apollo.

team in Zurich included twelve people from TCS (seven people from the Quartz team and five people from the Front-End team).

**Project 2: Dresdner**

The Dresdner project involved the implementation of Quartz at the Dresdner RCM Global Investors bank in San Francisco, which specializes in investment and e-commerce. Implementation of Quartz at the Dresdner bank started in July 2001. This project included implementation of Quartz as a front-office, integration with the local system and the development of several new components, such as securities trading, portfolio management, and communication manager, as illustrated at Figure 31.

**Figure 31: Dresdner project: Investment and e-commerce bank (from TKS report <http://www.tks-teknosoft.com/references/tbs/DresdnerNEW.pdf>)**



In terms of global distribution, the Dresdner project involved people from the Quartz group based in Gurgaon (offshore) and the customer site in San Francisco (onsite). At the time of data collection the offshore Quartz team in Gurgaon involved six people and the onsite team in San Francisco included eight people who had relocated from Gurgaon.

### **7.1.3 BACKGROUND OF THE SOFTWARE TEAM**

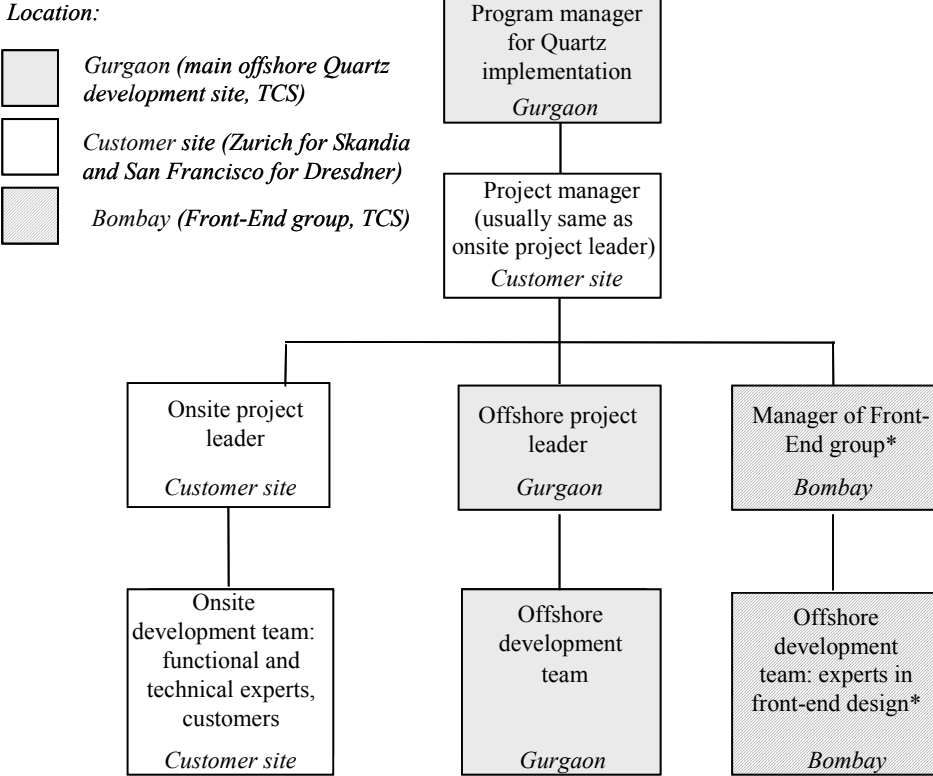
#### **7.1.3.1 WORKING EXPERIENCE IN A GLOBALLY DISTRIBUTED ENVIRONMENT**

From the first implementation in 1998, all implementations of Quartz took place in a globally distributed environment that included at least two locations: a customer site and the main development site of TCS in Gurgaon, where Quartz was customised (existing components were modified and new ones developed) to satisfy the requirements of a specific customer. Therefore, people in the Quartz group were used to working in a globally distributed environment.

#### **7.1.3.2 ORGANIZATIONAL STRUCTURE OF THE SOFTWARE TEAM**

The typical project organization of a Quartz implementation consists of an *onsite* team at the customer location and *offshore* teams at the development centres of TCS. A typical organizational structure of a Quartz implementation project is presented in Figure 32.

**Figure 32: Typical organizational structure of the Quartz implementation project: Skandia and Dresdner projects**



\* The Front-End group was involved in the Skandia project; it was not involved in the Dresdner project

***Interactions with a customer***

It is important to note that onsite and offshore teams consist of people from TCS only; they work closely with customer’s representatives. In general, interactions between a Quartz implementation team and a client are formal to a great extent. Customer’s representatives usually interact with an onsite team: in particular, the gap analysis stage and final user acceptance testing (see Figure 29) require many

interactions. Furthermore, customer's representatives fly to Gurgaon to participate in pre-acceptance tests which are done offshore.

TCS has formal procedures in place that help to capture a customer's requirements and to achieve mutual understanding and agreement (between TCS and the customer) regarding the scope of a Quartz implementation project. For example, technical, security and infrastructure requirements are agreed upon with the customer to avoid ambiguity in the future: 'one way is that the customer should have our high level design, standards and templates reviewed in the beginning and signed off' (Kumar Krishna, Manager of the Front-End group).

## **7.2 DATA COLLECTED**

Data was collected from a variety of primary data sources, which included: (i) interviews; (ii) internal project and company documents, and external reports and press releases; (iii) direct observations in Gurgaon and Zurich offices, i.e. one week in the Gurgaon office (early March 2002), and one day in the Zurich office (late March 2002); and (iv) informal conversations with managers and software engineers. Table 9 summarizes the names of interviewees, their roles, locations (Gurgaon, Zurich, San Francisco or Bombay site), and details of interviews and other communications for data collection purposes (roles are correct for March 2002). Furthermore, the case study by Alexandersen et al. (2003), which described the Skandia project from the strategic perspective of the client (Skandia Group), was used as a secondary data source.

**Table 9: TCS: Interview and data collection details**

<b>Name</b>	<b>Role</b>	<b>Location</b>	<b>Interviews and other communications for data collection purposes</b>
Sanjay Bhanot	Executive manager for Quartz	Gurgaon	<ul style="list-style-type: none"> <li>• 2 interviews at Gurgaon office on March 4 and 6 2002</li> </ul>
Sanjay Srivastava	Delivery manager for Quartz	Gurgaon	<ul style="list-style-type: none"> <li>• 2 interviews at Gurgaon office on March 5 and 8 2002</li> <li>• Follow-up by email with clarifications and additional information</li> </ul>
Sunil Singh	Offshore project leader for Dresdner	Gurgaon	<ul style="list-style-type: none"> <li>• 2 interviews at Gurgaon office on March 5 and 7 2002</li> </ul>
Sandeep Kumar	Project manager and onsite project leader of Dresdner	San Francisco	<ul style="list-style-type: none"> <li>• Phone interview on March 7 2002</li> </ul>
Bala	Software engineer (team of Sunil Singh)	Gurgaon	<ul style="list-style-type: none"> <li>• Interview at Gurgaon office on March 7 2002</li> </ul>
Pankaj Khurana	Offshore project leader for Skandia	Gurgaon	<ul style="list-style-type: none"> <li>• 2 interviews at Gurgaon office on March 5 and 6 2002</li> <li>• Follow-up by email with clarifications and additional information</li> </ul>
Sourin Som	Software engineer (team of Pankaj Khurana)	Gurgaon	<ul style="list-style-type: none"> <li>• Interview at Gurgaon office on March 6 2002</li> </ul>
Nitin Sironi	Technical consultant (former technical architect for Skandia)	Gurgaon	<ul style="list-style-type: none"> <li>• 2 interviews at Gurgaon office on March 6 and 8 2002</li> </ul>
N.G. Subramaniam	Vice President	Gurgaon	<ul style="list-style-type: none"> <li>• Interview at Gurgaon office</li> </ul>
Ashvini Saxena	Technical architect and team leader for Skandia	Zurich	<ul style="list-style-type: none"> <li>• Interview at Zurich office on March 28 2002</li> </ul>

Tuhin Sengupta	Software engineer (team of Ashvini Saxena)	Zurich	• Interview at Zurich office on March 28 2002
Krishna Kumar	Manager of Front-End team (head of team in Bombay)	Zurich	• Interview at Zurich office on March 28 2002, during his visit to Zurich
Rik Biswas	CIO Skandia Bank	Zurich	• Interview at Zurich office on March 28 2002
Rajan Bhatia	Project manager and onsite project leader for Skandia	Zurich	• Interview at Zurich office on March 28 2002

The empirical investigation included visits to TCS offices in Gurgaon and Zurich, and a phone interview of the onsite project manager for Dresdner located in San Francisco, and consisted of the following stages:

- First, initial contact and arrangements for data collection were done with the help of Girish Ramachandran, marketing manager for TCS in The Netherlands. He connected my supervisor Prof. Kuldeep Kumar with the TCS and the Quartz teams.
- Then, in March 2002 I visited the TCS office in Gurgaon where the first round of data collection took place: I conducted interviews and collected relevant documents. The interviews reflect on the whole period of the Quartz implementation, from the beginning of the implementations (autumn 2000 for Skandia and summer 2001 for Dresdner) until March 2002 when both projects approached completion.
- Next, late March 2002, a second round of data collection took place. I visited Skandia office in Zurich where the onsite Quartz team was located and conducted interviews.
- Finally, in September 2004 I collected the latest available financial reports of TCS, and press releases about Quartz.



## 7.3 HOW TCS ORGANISES AND MANAGES GD CBD: ANALYSIS AND RESULTS

In this section the analysis and results of the TCS case study are presented and discussed. First, managerial practices perceived as important in GD CBD are presented (Section 7.3.1). Then, the contribution of these practices to success in GD CBD is assessed and illustrated by empirical evidence from interviews (Sections 7.3.2-7.3.4).

### 7.3.1 TCS CONCEPT MAP

Data collected in TCS was analysed following the approach described in Chapter 4 (Section 4.5). In the TCS case in total 20 managerial practices were perceived by interviewees as important for success in GD CBD. During data analysis these practices were classified into groups that focus on different aspects of the management of GD CBD, in accordance with four factors suggested in the theoretical lens (Figure 15). Furthermore, one more factor emerged from the data, which is *Components management*.

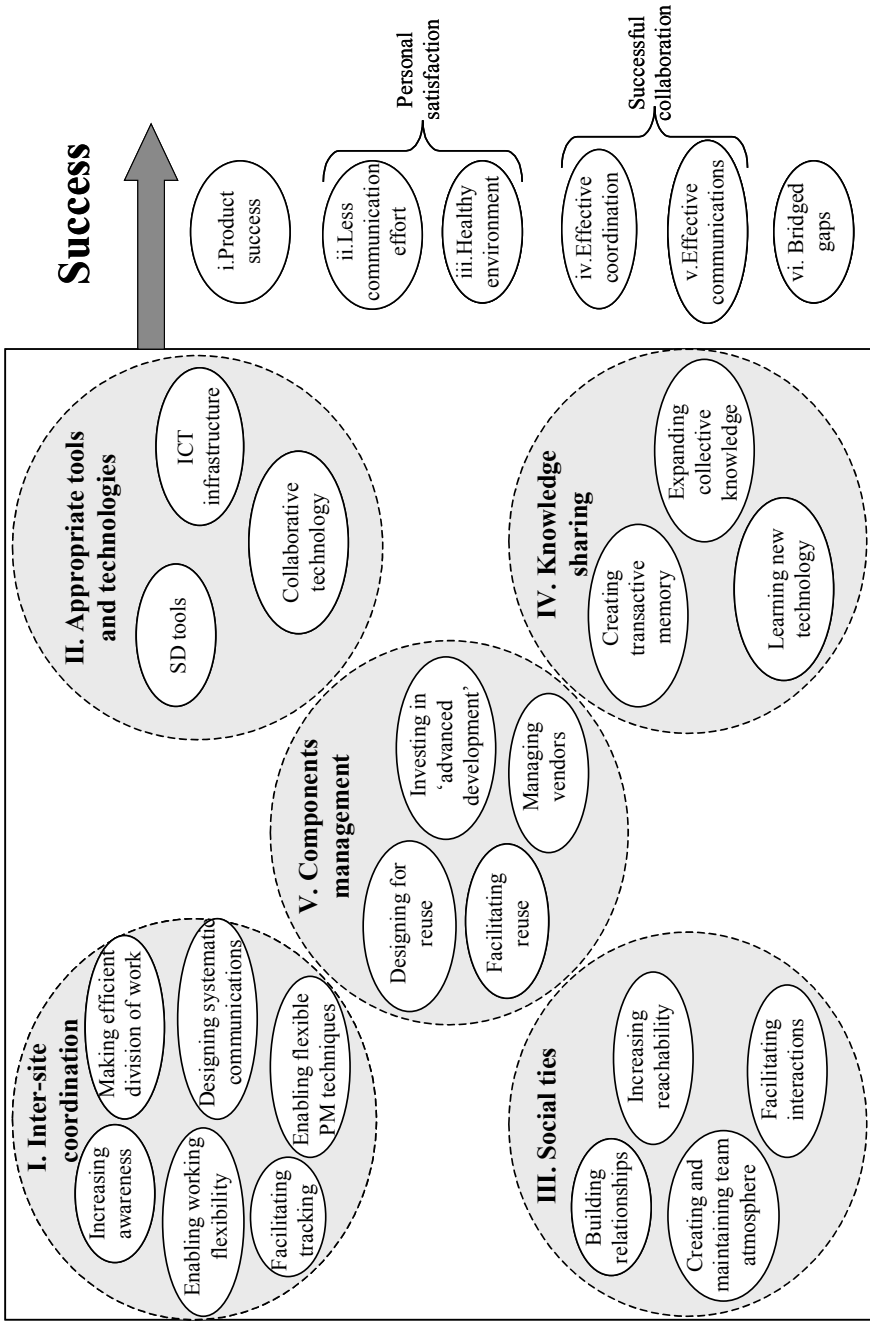
The managerial practices are classified into five groups according to the five factors and presented in the form of the concept map in Figure 33<sup>34</sup>.

In addition to the managerial practices, the TCS concept map contains categories representing evidence of success, which are (i) *product success*; then, personal satisfaction represented by two sub-categories (ii) *less communication effort* and (iii) *healthy environment*; successful collaboration represented by two sub-categories (iv) *effective coordination* and (v) *effective communications*; and the last category is (vi) *bridged gaps*. These categories and sub-categories are similar to those identified in the LeCroy and SAP cases.

---

<sup>34</sup> The concept mapping approach is explained in Chapter 4, section 4.5.1

**Figure 33: How TCS organises and manages GD CBD to be successful**



In the following sections the contribution of the five factors and corresponding managerial practices to success in GD CBD in SAP is illustrated and discussed using three types of data presentation (explained in ‘Within-case display’ in Section 4.5.1 and in Section 4.7).

### **7.3.2 FACTORS AND MANAGERIAL PRACTICES THAT CONTRIBUTE TO SUCCESS: CAUSAL RELATIONSHIPS**

The frequency of instances in which managerial practices were linked by interviewees with success is presented in Table 10<sup>35</sup>. As explained in Section 4.5.1, the numbers presented in the table give some indication of the strength of each relationship: the higher the number the stronger the relationship is.

---

<sup>35</sup> Table 8 is a *conceptually clustered matrix*, explained in Chapter 4, section 4.5.1: it shows the contribution of managerial practices and potential success factors to categories of success.

**Table 10: Contribution of managerial practices to success at TCS**

Managerial practices	Product success	Personal satisfaction		Success. collabor		Bridged gaps
		Less communication effort	Healthy environment	Effective coordinat.	Effective communic.	
I) Inter-site coordination	=>			=>	=>	=>
1 Increasing awareness	2			4		
2 Making efficient division of work	3			3	1	
3 Enabling working flexibility	1				1	1
4 Facilitating tracking of bugs and dev. tasks				3		
5 Enabling flexible PM techniques	3			1		1
6 Designing systematic communications	1			3		
II) Appropriate tools and technologies	=>			=>		
7 Software Development tools	2			3		
8 ICT infrastructure	1			1		
9 Collaborative technology						
III) Social ties	=>	=>	=>	=>		
10 Building relationships	2	1	2	1		
11 Increasing reachability	1					
12 Creating and maintaining team atmosphere	3			1		
13 Facilitating interactions	2			1		
IV) Knowledge sharing	=>	=>		=>	=>	=>
14 Creating transactive memory among teams	1			3		
15 Expanding collective knowledge of the team	4	1		1	2	2
16 Learning new technology						
V) Components management	=>			=>		
17 Designing for reuse	4					
18 Investing in 'advanced development'						
19 Facilitating reuse						
20 Managing vendors	2			4		

From Table 10 it follows that managerial practices that were most often explicitly connected to success by interviewees are inter-site coordination practices, in particular *increasing awareness* by ensuring continuous information flows between dispersed teams, and *designing systematic communications*; and knowledge sharing practices, such as *creating transactive memory* and *expanding collective knowledge*. These practices contributed to the majority of categories of success. Furthermore, *managing vendors* was important to achieve effective coordination and product success.

### **7.3.3 MANAGERIAL PRACTICES: DESCRIPTION AND EVIDENCE**

In this section managerial practices perceived by interviewees as important for success in GD CBD are described and illustrated using quotations from interviews (a detailed description of all managerial practices is included in the Glossary of Managerial Practices in Appendix 4).

#### **(I) Inter-site Coordination in GD CBD: managerial practices**

Interviewees stressed the need for inter-site coordination between onsite and offshore teams. This included the following practices:

##### **1. Increasing awareness**

The interviewees indicated that increasing awareness of (i) what is going on in the project at the remote location, and (ii) what everybody is working on, are important for success.

(i) Interviewees mentioned that it is important for offshore locations (Gurgaon and Bombay) to be aware of what is going on at an onsite location, about the development environment and technical infrastructure at the onsite location, to be able to visualise what is happening when a problem occurs, and to solve the problem. For example:

- Krishna Kumar (manager of the Front-End team) recognized that a structured approach is needed to manage globally distributed teams. He summarised lessons learnt from his experience in distributed development projects and created a document entitled ‘Lessons Learnt’ to serve as a guidelines for such projects. Some of the lessons learnt relate to the need to increase awareness, for example<sup>36</sup>:

The critical tasks should be graphically displayed in a chart for everybody to see. [...] There should be transparency of the processes, issues and problems faced in the development with the client.

- Sunil Singh explained: ‘When a project is being established, proper ground work includes that everything should be conveyed and project information shared among team members’.

(ii) Furthermore, it is important to build awareness of tasks that other team members are doing:

It's not that only one person can do a job, otherwise, if one person doesn't come in, we won't be able to work without him/her. So we try to overcome this by making each and every team member aware of nearly all the things which are happening (Sunil Singh).

Building awareness expands the collective knowledge of a dispersed team.

---

## **2. Making efficient division of work**

---

Efficient division of work is important: it involves principles that software managers follow to divide work between onsite and offshore locations, and to divide specific assignments (tasks) and responsibilities between individual teams at each location.

---

<sup>36</sup> More ‘lessons learnt’ summarized by Kumar Krishna are presented further in this chapter as quotations from the interview with Kumar Krishna.

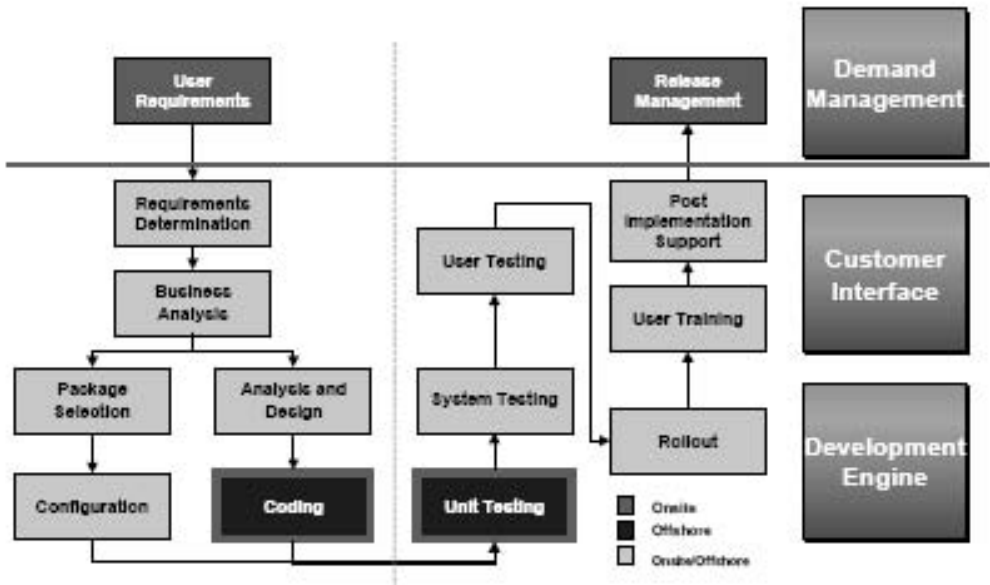
(i) Division of work between onsite and offshore teams, and the number of people at onsite and offshore locations, varies at different stages of a Quartz implementation project. Usually, there are more people onsite in the earlier stages of the project, when close interactions with customers are required, and in the later stages during final integration and end-user testing. During the design and construction stages some people from the onsite team relocate to offshore locations to work on the development of new components and modification of core Quartz components (i.e. people from Quartz group go back to Gurgaon and people from Front-End group go back to Bombay). For example, the number of people at the offshore location in Gurgaon during the design and construction stage of the Skandia project was about 35 people, which was reduced to 6 people during the final stages.

Figure 35 illustrates the Skandia onsite-offshore delivery model adopted by TCS, which combines steps conducted onsite, offshore, and a combination of onsite and offshore. Transferring some steps to offshore allowed TCS to take advantage of cost, quality, and advanced delivery capabilities infrastructure (Alexandersen et al. 2003)<sup>37</sup>.

---

<sup>37</sup> The onsite-offshore delivery model is explained in greater details in Section 2.4.6, following Figure 13.

**Figure 34: Skandia project: onsite-offshore delivery model (adopted from Alexandersen et al. 2003)**



The main strategy of TCS is: ‘maximize work to be done offshore: keep onsite as little as possible people because this adds costs’ (Sandeep Kumar). Typically, an onsite team sends requirements offshore ‘because the expertise and major source code are here [offshore, in Gurgaon], and mainly because of the expertise, it is quicker and easier to work here’ (Pankaj Khurana).

(ii) Moreover, role continuity and project ownership are important for successful implementation of Quartz. Role continuity implies that people who are doing gap analysis (between requirements and available components) are also doing the development, because they understand the requirements. For example, in the Dresdner project:

The person who had done the requirement study there [onsite, in San Francisco] for the trading system came back [to Gurgaon], and he



started leading the trading system development team, three people came back with him (Sunil Singh).

Despite the fact that the project is transferred between onsite and offshore locations at different project stages, ownership of the project stays with the same team: team members are transferred between onsite and offshore together with the project, and work continuously on the same project / components. This setting helps to ensure that customer requirements are understood, captured and implemented in the product. Kumar Krishna elaborates on roles and responsibilities:

- ‘Roles and responsibilities of a team need to be clearly specified. Proper back-up is needed for each responsibility, to accommodate release and movement of personnel across geographies’
- ‘Decentralisation and proper delegation of work are needed to avoid bottlenecks and time lags’.

(iii) Furthermore, the composition of the remote teams is important, in particular of the onsite team. The onsite team is composed of experts who provide expertise in areas required at a customer location, and involves technical, functional and support roles:

Technical people look at technical and architectural issues, functional people look at the functionality and the development of the functionality. Support people handle the configuration management, the groupware, the other various tools which are being used by the team (Sanjay Srivastava).

The onsite team has technical responsibilities and is responsible for implementation of customer requirements. This team provides technical support for teams in the main Quartz development centre in Gurgaon and the Front-End development centre in Bombay.

(iv) Additionally, the division of work between team members at dispersed locations is done according to their skills (expertise):

Between us [offshore] and our onsite team we say ‘we’ll do this portion of the job because we have more competent people here who can look at this part, and you can look at that portion of the job’. It’s mutual communication (Sunil Singh).

---

### **3. Enabling working flexibility**

---

The interviewees suggested that working flexibility, in terms of providing flexible working conditions, such as mobile phones and computers that allow working from home early and late in a day, and providing flexible working hours, is important for success.

Having flexible working hours (e.g. starting earlier in Zurich and later in Gurgaon) makes it possible to increase the overlap in working hours between locations so that remote teams can collaborate in real-time. Sometimes, in particular during end-user testing when customers are closely involved in the implementation process, team members at the offshore location in Gurgaon stay in the office until late to be able to provide support to the onsite team that is working closely with the customer.

Out of working hours remote team members can contact each other at home by (mobile) phone, as often happens in the Dresdner project, e.g. when Sunil Singh (in Gurgaon) needs to contact Sandeep Kumar (in San Francisco) for clarifications when Sandeep Kumar is already at home, because of the 13.5 hours time difference.

---

### **4. Facilitating tracking of bugs and development tasks**

---

Interviewees at TCS suggested the importance of tracking bugs and development tasks.

- *Tracking of development tasks*

It is very important to be able to track development tasks during an ongoing project, in particular while working around-the-clock. Sunil Singh and Sundeep

Kumar (offshore and onsite project leaders of Dresdner respectively) explained that during the late stages of a project they often work around-the-clock by sending tasks back and forth between Gurgaon and San Francisco (some of these tasks are fixing bugs). For tracking they use Excel spreadsheets that they update every day, and email an updated file to each other in turns (once Sunil and once Sandeep).

Interviewees mentioned that for each component there is a need to know who developed it, because if the component needs to be modified, typically, there is a need to consult with the developer who originally wrote a particular code of the component; or even delegate the modification to him/her, if possible. Therefore specifications of each component should include the name of a person who developed it.

- *Tracking of bugs*

The Quartz group uses a PVCS Tracker software tool to support the tracking of bugs. However, during the last stages of Quartz implementation, when bugs need to be fixed very quickly, often team members avoid the procedure of reporting bugs in the system and use the help of remote counterparts in a non-official manner.

## **5. Enabling flexible Project Management (PM) techniques**

Interviewees suggested that flexible PM techniques are important to accommodate complexity and everyday dynamics. They include:

- On a macro level: Planning of major project phases

Krishna Kumar stated that there is a need for a ‘unified project plan at a reasonable detailed level and not merely at a higher level, clearly stating the dependencies; clear milestones need to be marked’.

- On a micro level: Flexible and not too detailed planning.

There is a need for flexibility in accommodating changes:

Distinction between clarifications/corrections and changes should be made and agreed upon. Changes can never be avoided. If every change is evaluated and postponed for future implementation the final product will not satisfy users' expectations. One should know where to draw a line (Krishna Kumar).

## **6. Designing systematic communications**

Systematic communications are considered important for success. Design of systematic communications includes:

(i) Scheduling systematic and frequent communications, such as regular teleconferences between software managers in dispersed locations. Krishna Kumar described the situation as follows:

Project and Program managers have to regularly meet with the personnel to appraise the status of the project, to share management views where applicable, to discuss processes and why and how they are impacting the work, to discuss revision of plans, besides to motivate the team. This forum can also be used to address grievances. This should be a regular practice at the site where the team members are located and this should be percolated to the different geographies through various team leads to their members. Mails are not sufficient.

Typically, onsite and offshore leaders communicate by phone at least twice (and sometimes three or four times) a day:

- For the Skandia project it happens (1) when the team in Gurgaon starts their working day (which is midday in Zurich), and (2) before the team in Zurich leave home (which is midday in Gurgaon).
- For the Dresdner project the first teleconference takes place when the offshore team starts its day (at that time the onsite team in San Francisco is about to leave home), and the second teleconference takes place at the end of the day for the Gurgaon team (when the team in San Francisco starts their day).

Usually, not only onsite and offshore project leaders but also team members participate in the teleconferences. I attended one of the teleconferences as an observer. Issues discussed during that conference covered progress update, handover of work from one team finishing their working day to another team starting their working day, and clarifications.

(ii) Communicating directly to reach an appropriate person, i.e. avoiding hierarchy in communications. For example, Krishna Kumar stated: ‘Proper communication at all levels through appropriate means, including regular telecons, update of statuses, is key to success’.

## **(II) Appropriate Tools and Technologies in GD CBD: managerial practices**

Managerial practices related to tools and technologies identified in TCS as important in GD CBD are as follows:

### **7. Software Development (SD) tools**

Software development tools include tools for the development and management of components, configuration and version management tools, and tools for testing and tracking bugs, such as<sup>38</sup>:

- Master Craft Tool set
  - ADEX - Repository
  - QDE-IF Process framework
  - Generators and translators
- MS-Access for Issues registration & resolution
- SQA Robot - for regression testing
- PVCS Tracker - for defect logging, tracking & analysis
- PVCS Version Manager - for configuration control

<sup>38</sup> Based on TCS internal documents

- RoboHELP - for Help and user manual
- SQL Lab, TOAD for SQL Analysis & Optimization
- Other in-house tools for Performance modeling, Costing etc.

Furthermore, the following tools are used for project management and Quartz documentation:

- MS project, Excel - for Project planning & monitoring
- MS Office - for documents, Lotus Notes for email and internal communication

In order to support CBD in a globally distributed environment SD tools need to provide the following capabilities.

---

**Standardization of tools across locations** At TCS methods and tools are standardised across locations in two ways.

**locations** First, all development teams working on Quartz use the same tools and methods and follow same processes: this helps to ensure quality of processes:

In a distributed development environment, we need to clearly identify the quality processes to be followed and ensure commonality in the compliance of such processes. For example: common processes and tools for bug tracking, configuration management, release management, impact analysis, change management (Krishna Kumar).

Second, for each implementation of Quartz, customer-specific methods, tools and processes are standardized across globally distributed onsite and offshore teams.

Quartz is concerned with banking, where a lot of data and information are confidential, thus TCS cannot have access to the client's actual system for the final (end-user) testing. To overcome this problem, the offshore development team (in

---

---

Gurgaon) creates a development environment that replicates the one at the customer site. This way, onsite and offshore development teams work together: the onsite team at the customer site working in a real-life environment can delegate work (in particular, bug fixing during testing) by sending the actual code to the offshore team in Gurgaon which can continue working in the replicated environment. For example, Sunil Singh explained about the Dresdner project:

All the code, everything which they [onsite team] have is in synch with what we have here [offshore, in Gurgaon]. We work in parallel: we send them the source code and they integrate it into the infrastructure before delivering it to the client.

---

**Centralisation  
of tools**

There is one central repository, a central server in the main development centre in Gurgaon, where the source code is maintained; therefore, if the onsite team changes source code, the team send the source code to the headquarters where it is integrated into the main repository.

However it is difficult to work in two development environments in parallel because the source code needs to be coordinated manually. In a single development environment the code is coordinated automatically by a ‘baseline’ mechanism that makes it possible to check code out and in, so that a chunk of code can be checked out only once, and it is considered as ‘frozen’; and until it is checked back in, nobody can work on the same chunk of code. However, when there are two development environments, onsite and offshore, the ‘baseline’ needs to be maintained manually:

I’ll check out our source code and send the code to the

---

---

onsite team to work on it. Then they send the changed code back to me and I'll check it in (Sunil Singh).

Furthermore, documentation needs to be centralised:

Multiple documentation should be strictly avoided. Every additional work / change in scope should be coming in as Change Request Specification (CRS). All clarifications, interpretations should be documented in a common place. Too much information should not be cluttered (Krishna Kumar).

---

**Standardization of methods across locations** Standard procedures are developed to ensure that specifications written by the onsite team are understood correctly by the offshore team. As Pankaj Khurana explained:

We have set procedures for defining the requirements. If people follow the procedures, then the things become very easy to interpret or understand.

However, standard procedures are not always followed:

Usually people take shortcuts and explain over the phone, instead of writing a complete specification and emailing them to offshore, and it is a bit dangerous. Because, for example, after six months the specification document becomes very important, but people who did the change are not there, they should have given specifications when they did the change (Pankaj Khurana).

---

**Creating guides that explain procedures and methods, and project template documentation** TCS has a set of standards and guidelines for the various phases and deliverables for all project life cycle activities for Quartz implementation. It includes:

- Procedural standards that provide the team with a set of practical tools and techniques with guidelines on when and how to use them
- Documentation standards that provide the team with means

---



---

of preparing the identified tangible deliverables.

For example, procedural standards include the Quartz Implementation Methodology, and a Program Development Process for Quartz system. During data collection in Gurgaon I obtained access to these documents: however, they cannot be included in the thesis for confidentiality reasons.

Documentation standards contains different templates for Quartz implementation projects. A standard Quartz documentation set includes the following templates (based on TCS internal documents):

- Project Documentation Set
  - Business Requirements Overview (BRO)
  - Business Requirements Specifications (BRS)
  - High Level Design Document
  - DB Design Document
  - Module Test Specifications
  - Product Acceptance Testing Specifications
- User Documentation Set
  - Online Help
  - User’s Manual
  - Installation Manual
  - Operations Manual

For specific Quartz implementation project these template documents are filled in and, if necessary, modified.

---

**Developing  
tools in-house**

In TCS, the majority of SD tools are built in-house. Some of these in-house developed tools are available on the market as off-the-shelf software packages, e.g. the Integrated Project Management System, which is used for project management.

---

---

## 8. ICT infrastructure

---

ICT infrastructure needs to support the following capabilities:

---

**Quick and easy connectivity across locations** Quick and easy connectivity between locations is necessary to send the source code back and forth between onsite and offshore locations. TCS uses a ftp server to transfer the code: ‘we don’t have any problems, we can send anything via the ftp server’ (Sunil Singh).

---

**Web access** Web access is needed for version and configuration management, because ‘version control on the Web would solve the problem of manual checking in and out of source code’ (Sandeep Kumar).

Furthermore, Sandeep Kumar suggested that the Excel spreadsheets they use to coordinate transfer of tasks between remote teams in a follow-the-sun manner, needs to be Web-based (instead of sending it back and forth between dispersed locations, as he and Sunil Singh were doing).

---

---

## 9. Collaborative technology

---

The following are collaborative technologies used by TCS team to collaborate successfully over distance:

---

**Phone and teleconferencing** A phone is used on a regular basis for onsite-offshore coordination: for example, onsite and offshore project leaders and managers use phone for updates, clarifications, and resolving issues.

---

---

**Application Sharing** Application sharing is used often for bug fixes, for example to show conditions of a system failure. As mentioned earlier, for security reasons, typically, the offshore team does not have access to a customer system. Therefore, the use of application sharing between onsite and offshore teams is limited. In some cases, instead of using application sharing, the onsite team needs to send a source code to the offshore team and/or describe the problem or bug by phone or email.

---

**Videoconference** Videoconferencing is used mainly between executive managers and with customers, less often between project leaders and managers.

---

**Email** Email supports low priority tasks and issues, and tasks that cannot be completed in real-time because of time-zone differences. Email is sometimes used for sending changes in source code:

If it is a very minor change like a small portion with two lines changed in the code, we send the code through email (Sunil Singh).

---

**Intranet** The Quartz group has access to TCS Intranet, which has a repository for Quartz group where internal documents and other relevant information are posted.

---

### **(III) Social Ties in GD CBD: managerial practices**

According to the opinions of interviewees, rapport and trust contribute to success. In the Quartz group trust and rapport between members of onsite and offshore teams were developed to some extent because the majority of them have worked

together in a co-located environment on the development of Quartz and/or knew each other before re-locating to a customer location (Zurich for Skandia and San Francisco for Dresdner). Following are the managerial practices identified as important to further develop trust and rapport between remote counterparts.

---

## **10. Building relationships**

---

Interviewees consider having good relationships between remote counterparts very important for success. The following quote illustrates the importance Quartz managers give to building relationships between team members:

In the last few days you've been here and you have seen the environment that we are working in [time pressure, customer-driven: every day new tasks and changes coming from the onsite team]. In such an environment, I think, the most important person is the actual person who does the work, I am just a facilitator here. The day we started the project we agreed - and it was a conscious and unconscious decision - that everybody has to work together to make this project successful, and they have to know some portion of what other team members are working on. If there is friction between team members, it cannot work. So I tried to make that situation correct between each of these people (Sunil Singh).

---

## **11. Increasing reachability**

---

Increasing reachability, i.e. being able to reach the right people at a remote location, is important for successful Quartz implementation. Usually, members of the Quartz group know whom to contact at a remote location: they know the area of expertise of each other from working on previous projects, and because the majority of members of the onsite team spend some time at the offshore location during design and construction stages.

The main difficulties in reaching the right people are caused by time-zone differences, in particular in the case of the Dresdner project, where regular

working hours of onsite and offshore teams do not overlap because of a 13.5 hours time-zone difference. To deal with time-zone differences, members of Quartz group have mobile phones and can reach each other by mobile or home phone when their working hours do not overlap. Furthermore, if a counterpart from an onsite/offshore team is needed during his night-time, sometimes Quartz team members call an expert involved in a different project with whom working hours do overlap.

Interviewees suggested that, because the Quartz team members can easily contact each other at any time of a day, they can work faster and utilise time-zone differences to work around-the-clock. While working with clients and vendors (suppliers of third-party components and services to TCS), with whom reachability is limited to formal contacts during official working hours, completing the work takes longer:

With other companies which are working with us, our vendors, we have to be very very formal in the sense that we can contact them only during office hours and/or they can contact only the official support people. They go from one professional service to another professional service and, therefore, it takes a long time for them to actually arrange for people to be available to solve a problem. Within Quartz we can actually call up anybody whom we know at any point in time to get some assistance, even if we don't know somebody, if he's recommended by someone else, then we can call up and get assistance immediately. It is a very considerable difference (Ashvini Saxena).

In general, in the Indian culture it is considered normal that one can approach one's counterpart outside working hours, as opposed to many European cultures, e.g. Dutch, Swiss, German, where it is not common to contact somebody about work outside of his/her working hours. For example, as Ashvini Saxena explained:

If I am facing some problem in my project with respect to a particular area, I can go back home at 10 o'clock at night and knock on the door of a person who might be able to help and just ask him to help me out.

## **12. Creating and maintaining team atmosphere**

Creating and maintaining a team atmosphere among onsite and offshore teams is important for success. In TCS, people involved in the Quartz development and all Quartz implementations consider themselves as the ‘Quartz family’ with their own ‘Quartz culture’, and ‘Quartz language’. People involved in Quartz implementation talk about themselves as ‘our own people’: they do not distinguish between dispersed onsite and offshore teams ‘we’ versus ‘they’.

Krishna Kumar emphasized the importance of the team members, team atmosphere and motivation for success:

- ‘Members are the key to the success. They should be well treated, accommodated, well informed on the schedules and plans for each task. Tasks and schedules should not be committed to a client without acknowledgment from the team responsible for development’.
- ‘Members should not be expected to work over weekends/ holidays. They should be given proper intimation if they are needed to. The schedule should take this into consideration’.

## **13. Facilitating interactions**

Facilitating interactions between people at remote locations is important. In TCS members of onsite and offshore teams interact frequently. First, onsite and offshore project leaders work very closely and interact on a daily basis (as described earlier in the ‘design systematic communications’ practice). Second, members of onsite and offshore teams have an opportunity to meet in person during the design and construction stages when most of the onsite team members come back to join the offshore team. Furthermore, some of them have interacted during earlier projects and/or the Quartz training program that is compulsory for anybody joining the Quartz group.

#### **(IV) Knowledge Sharing in GD CBD: managerial practices**

Interviewees consider knowledge sharing as contributing to success: in particular, building up collective knowledge through shared experiences, and creating transactive memory among team members at dispersed locations (Zurich, Bombay and Gurgaon for the Skandia project, and San Francisco and Gurgaon for the Dresdner project).

In the global software team of TCS transactive memory and collective knowledge were developed before the project started. In particular, the collective knowledge is very broad, because all team members have the same cultural background (developers in Gurgaon, Bombay, Zurich and San Francisco are all Indian), and collective knowledge to a great extent is based on national culture (Baumard 1999). Furthermore, in TCS team members also had collective technical knowledge about Quartz from Quartz-related training and their own experience in Quartz development and implementation.

Following are managerial practices seen as important for knowledge sharing between remote team members, supported by quotations from interviews.

#### **14. Creating transactive memory among dispersed team members**

Creating transactive memory among team members located onsite and offshore, and among people involved in Quartz group (which includes all Quartz implementation projects TCS is involved in) is considered important for success.

In the Quartz group there are a number of activities, such as training programs, that facilitate interactions among members of Quartz group through which team members get to know each other and create transactive memory.

The following quotes illustrate the existence of transactive memory at the studied team:

- ‘I am involved since the start, so I know each team member and everything which has been done’ (Sunil Singh).

- Sandeep Kumar, on-site manager said about offshore team: ‘I know team members very well, know their strengths’.

## **15. Expanding collective knowledge of the dispersed team**

Expanding collective knowledge of the dispersed team is important for success. In addition to the knowledge of national culture that all team members possess, team members need to have collective knowledge of the overall product (beyond a specific area an individual team member is working on), which includes (i) cross-functional knowledge, (ii) understanding of logic (changes) in the evolving product, and (iii) common language / terminology

(i) Cross-functional knowledge provides the team with flexibility to accommodate everyday dynamics and uncertainties by reducing dependencies on specific team members. For example, if needed, team members can ‘replace’ each other:

Each and every team member is aware of nearly all the things which are happening, the whole team has a basic knowledge about everything. Usually they work on their own specific code areas, and only in circumstances when the other person is not available, they would work on the other areas. But to make it easier for them to work on the other areas, they have to have the basic understanding of that area (Sunil Singh).

ii) People in TCS are convinced that it is important to understand the logic behind the code and changes which have been made in the code. For example, Sunil Singh explained that when his team in Gurgaon and onsite team in San Francisco work around-the-clock by sending code back and forth, they also send descriptions of the changes ‘so that the onsite team understand the changes and don't have to spend time understanding what changes we have made’ (Sunil Singh). Describing changes made by remote counterparts helps to expand the collective knowledge of the dispersed team members, sharing understanding of the evolving product.



Furthermore, Pankaj Khurana explained that it is important to describe the logic behind the code so that in case a component is handed over from one developer to another to continue working on it, or a component would need to be modified in the future, anybody (and not just the person who wrote the original code) can continue working on and/or modify the component. Thus, by documenting the logic behind components the Quartz group externalise the tacit knowledge of individual team members and convert it into explicit knowledge (Nonaka and Takeuchi 1995) available for the whole Quartz group, for any Quartz implementation projects.

(iii) Furthermore, collective knowledge includes the use of common language / terminology between remote team members. As Pankaj Khurana described it:

We all speak Quartz language. It is a loss for us if somebody leaves Quartz because for somebody new it will take time to learn Quartz.

To utilise the collective knowledge of the people involved in Quartz, Pankaj Khurana explained: ‘people rotate within Quartz, not out of Quartz’.

---

## **16. Learning new technology**

---

Interviewees from TCS consider the learning of a new technology important for success.

In TCS, learning new technology is concerned with (i) learning the programming language and tools used for developing Quartz (e.g. the Master Craft Tools, described earlier in ‘SD tools’ practice), and (ii) learning theoretical principles and different business (financial/banking) functions included in the Quartz platform.

For the learning of Quartz and technologies used to develop it, TCS organises intensive courses in which team members from globally dispersed location all gather at one location, a training center at Trivandrum. For anybody joining the Quartz team attending this training program is compulsory.

## **(V) Components Management in GD CBD: managerial practices**

In addition to the four factors suggested in the theoretical lens, *Components management* emerged from the TCS data as a factor contributing to success.

Following are the managerial practices seen as important to ensure successful management of components. These practices are important in co-located CBD as well; however, they become more critical in a globally distributed environment.

### **17. Designing for reuse**

Interviewees consider that applying a design-for-reuse strategy is important for success. In TCS the main advantage anticipated from the CB Quartz architecture is to be able to reuse it in the long term for a number of clients. In order to maximise reuse across different Quartz implementations for different clients all over the globe, jointly with TKS, the TCS team invested time and resources to identify the most common requirements for banking and financial services. The analysis addressed issues such as (i) what components to develop (what functionalities are required that are common for all / a majority of potential clients), and (ii) what should be the granularity of components.

In each Quartz implementation the majority of components included in the Core Quartz platform (described in Section 7.1.2) are reused. However, since all projects are somewhat different, for each implementation some additional functionality has been developed. For example, in the Dresdner project Quartz was integrated with the client's system as a back-end system (while originally Quartz was developed as a front-end system). Sunil Singh explained:

We used a distinct structure of parts, which was already present: we just made minor variations to that, I can say 50-60% were reusable.

Furthermore, TCS exploits customer-specific components by adding them to the Quartz package so that they can be reused in future Quartz implementations. Following this approach, with each new Quartz implementation TCS increases the

variety of components / functionalities that TCS can offer to potential clients. For example, TCS implemented Quartz at Royal Skandia UK<sup>39</sup>, an insurance company, where Quartz was implemented as an investment engine. Quartz, originally developed as a banking application, had never been implemented in an insurance company before. Sanjay Srivastava explained about the changes that were made to Quartz:

A lot of changes were made to the basic Quartz system just to be able to integrate it with the insurance business. We had to build in a lot of things that deal with policy administration and policy distribution, which are not particularly bank products. This way typical insurance products were added to Quartz: they were released as the next version of Quartz.

The use of a CB architecture facilitated reuse of components across different Quartz implementations at different geographical locations.

---

## **18. Investing in ‘advanced development’**

---

Investing in advanced development was considered important by interviewees at TCS. Advanced development in TCS included cooperation with TKS: it was based on integrating core capabilities and knowledge of the two companies – the technical knowledge in developing advanced software products of TCS, and the business knowledge of financial processes, regulations and clients in Europe of TKS.

---

## **19. Facilitating reuse**

---

Interviewees from TCS indicated that facilitating reuse of knowledge and components across different Quartz implementations is important for success in

---

<sup>39</sup> Royal Skandia UK is a different company from the Skandia Bank Switzerland discussed in this thesis in the context of the Skandia project

GD CBD. There is a central role, *Quartz program manager*, who is coordinating all Quartz implementation projects across all dispersed locations. The Quartz program manager has an overview of all projects, i.e. he is aware of new components being developed for a specific customer and can facilitate the reuse of these components across different implementation projects.

Furthermore, to facilitate reuse of knowledge and components in a globally distributed environment people are rotated between onsite and offshore locations to bridge knowledge gaps between the two sites.

Moreover, by being involved in several functional or technical areas, people develop and extend their expertise. They develop cross-functional knowledge in these areas, and can apply this knowledge later when they move to other (subsequent) implementation projects.

---

## **20. Managing vendors**

---

Typically, Quartz implementation involves integration of Quartz components with the client system and third-party components (as described in Section 7.1.2). Therefore, it is important to manage vendors providing third-party components: selecting vendors, agreeing on specifications of the components (e.g. functionality and interfaces), deadlines for components' delivery. In particular, vendor management was very important for the Skandia project, where more than 25 vendors were involved in delivering components. In TCS it is a responsibility of the Quartz program manager to guide and coordinate work between all parties involved in the implementation project: onsite and offshore teams, and vendors of third-party components:

Vendor management is needed when you have software vendors: for example, buying security software from someone, or buying hardware from someone. So the Quartz program manager will not only look at what onsite and offshore teams are doing, but he will also look at what

vendors are doing and coordinating between the activities of all the vendors, all the interested parties (Sanjay Srivastava).

### **7.3.4 SUCCESS IN GD CBD: EVIDENCE**

This section presents evidence collected in interviews about the success achieved in the studied case. The evidence is presented according to the categories of success illustrated in the TCS concept map (Figure 33).

#### **i. Product Success**

The Quartz project was highly successful. In 2002 Quartz was recognized by the International Banking Systems (IBS) Journal as being among the best-selling banking systems. The IBS newsletter (March 2003) states:

Quartz from Tata Consultancy Services/TKS-Teknosoft did well. This is now sold across a relatively broad range of activities, including asset management. It took a fair while to make it to market but now looks proven and well rounded.

Since 2002, according to annual 2003 and 2004 IBS reports<sup>40</sup>, Quartz has remained among the top 25 best-selling banking systems.

Furthermore, as intended by Quartz development group, the CB Quartz platform was reused for a number of clients (see ‘designing for reuse’ practice in Section 7.3.3 for more information about the success of reuse in Quartz platform).

#### **Personal Satisfaction**

##### **ii. Healthy environment**

Sunil Singh (offshore project leader of Dresdner) describes the team atmosphere in the Dresdner project:

---

<sup>40</sup> The IBS Annual Sales League Tables are available from the web-site:  
[http://www.ibspublishing.com/sales\\_league\\_tables/league\\_tables.htm](http://www.ibspublishing.com/sales_league_tables/league_tables.htm)

Over a period of time (I have known them for around 6-7 months), I know what they feel and what they don't feel. And if somebody has problems at home and he wants to take leave, I try to go ahead and look for other people who can do his job. These things are managed very well within our team.

### **iii. Less communication effort**

Talking about his remote counterparts Tuhin Sengupta said:

They know Quartz and I know Quartz, so little things are easy to explain: 'you go there and you do this' - it's not difficult to explain.

## **Successful Collaboration**

### **iv. Effective coordination**

One of the most important issues in Quartz is ensuring that third-party components are delivered on time and according to specifications. Thus, the program manager facilitates the building of individual plans for participating vendors:

so that they deliver components when you need them, that one component of the software is delivered on time for the next component, e.g. the hardware is delivered on time for the software (Sanjay Srivastava).

### **v. Effective communications**

The following example illustrates that a division of work based on expertise of dispersed team members improves efficiency of work:

There are two areas in which we are working: trading and portfolio management. We have clearly-defined jobs: some people would be working only on trading and other people would be working only on portfolio management. They are very familiar with their area, with each and every line of the product, because they are writing it, they have developed the product, so they are very clever, they know what is there. So once we tell them 'this thing has to be changed', it doesn't take much time for them to understand and change it (Sunil Singh).

## vi. Bridged geographical, time-zone and cultural gaps

*Geographical distance* is not perceived as a problem:

For onsite and offshore teams geographical distance causes limited inconvenience, i.e. when the help of the offshore team is needed to fix bugs during end-user testing at the customer location. In such a situation the offshore team cannot access the customer system from a remote location for security reasons. However on a regular basis, team members communicate remotely using different types of communication media.

*Time differences* cause some problems, but usually is used as an advantage:

Interviewees explained that despite the fact that a time difference such as 13.5 hours (in the Dresdner project) is causing some difficulties (e.g. onsite project manager Sandeep Kumar is often contacted during late hours in the evening when he is at home), onsite and offshore teams can work faster and utilise time-zone differences working around-the-clock.

For example, when it is night-time in Gurgaon, instead of contacting offshore team members at night, the team in Zurich can get help from people involved in different Quartz implementation projects but whose working hours overlap with Zurich. For example:

We can get help from someone who is already in the office somewhere else, then they can help to solve a problem from there. And, if we are working until 8 or 9 o'clock in the evening, then it's already overlapping with office time in USA. So they can call us up and we can support them (Ashvini Saxena).

*No cultural differences* within the Quartz group:

Team members in all locations have the same cultural background: they are all Indian.

## 7.4 CONCLUSIONS

In this chapter the analysis and results of the TCS case study were presented and discussed. Managerial practices and quotations from interviews illustrating these managerial practices and their contribution to success were presented.

The results of the case study illustrate that interviewees considered four factors suggested in the theoretical lens, and the fifth factor (components management) that emerged from the data, as contributing to success in GD CBD.

In terms of managerial practices, *inter-site coordination* between onsite and offshore teams was effective and efficient: first, the main strategy that TCS followed to divide work was to do maximum work offshore and minimise work onsite; work was divided based on expertise. Second, in order to increase awareness and keep remote teams updated all the time, systematic communications were organised between onsite and offshore managers and developers. Moreover, ownerships of the work packages stayed with the same team: team members were transferred between onsite and offshore locations together with the work packages (components) they were working on.

In relation to *appropriate tools and technologies*, methods, tools and processes were standardized across globally distributed onsite and offshore teams. However full standardization and centralization of tools was not possible, as Quartz is concerned with banking, where much data and information is confidential. To overcome this problem, the offshore development team in Gurgaon created a development environment that replicated the one at the customer site.

Regarding *social ties*, in TCS trust and rapport between remote counterparts were developed before the projects started, because the majority of team members have worked together and knew each other before re-locating to onsite locations. Furthermore, the team atmosphere in Quartz group is remarkable: onsite and



offshore teams consider themselves as the 'Quartz family' with their own 'Quartz culture' and 'Quartz language'.

Moreover, the Quartz team members could work faster and utilise time-zone differences to work around-the-clock, because they could easily contact each other at any time of a day (approaching one's counterparts out of working hours is considered normal in Indian culture, as opposed to many European cultures, where work-related communications are limited to working hours only).

Concerning *knowledge sharing*, in TCS global team transactive memory and collective knowledge were developed before the projects started because all team members have the same cultural and technical backgrounds, and became the majority of team members knew each other. To facilitate knowledge sharing Quartz managers rotated people between onsite and offshore teams.

It is important to acknowledge how effective the *components management* was organised in TCS: first, in the Skandia project in which more than 25 vendors of third-party components were involved, coordination of all dispersed parties - onsite and offshore teams, and vendors - was centralized under the supervision of the Quartz program manager. Second, in order to maximise reuse across different Quartz implementations for different clients, jointly with TKS, the TCS team invested time and resources to identify the most common requirements for banking and financial services.

This chapter presented and discussed the TCS case study. In the next chapter the Baan case study will be presented and discussed.

## CHAPTER 8 CASE STUDY OF BAAN

*Technology comes to our rescue in working in a distributed environment*

(Venkat Rao, Product Manager, Baan)

*But is technology alone enough to succeed in a globally distributed environment? Probably not, as we can learn from the unsuccessful Baan E-Enterprise case where technology was in place but the rest, inter-site coordination, social ties and knowledge sharing, were lacking*

### 8.1 BACKGROUND

#### 8.1.1 BACKGROUND OF BAAN GLOBAL ORGANIZATION

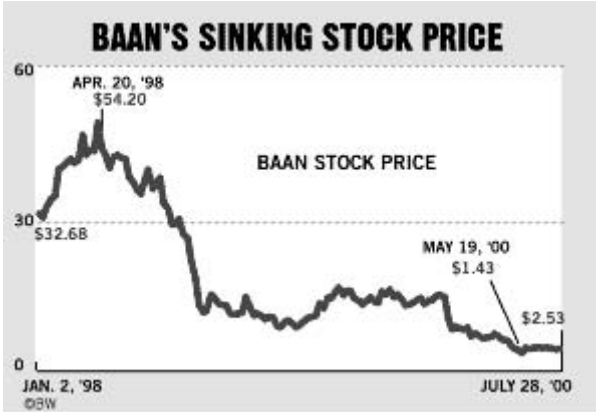
The Baan Corporation was created in 1978 by Jan Baan to provide financial and administrative consultancy services. A few years later his brother, Paul Baan, joined the company. Baan started to develop software packages, and in the mid 90s, with the emergence of the Enterprise Resource Planning (ERP) industry, Baan became one of the market leaders and biggest vendors of ERP software, competing with SAP, PeopleSoft and Oracle. In the mid 90s Baan opened several development centres in different countries: the main sites were in Hyderabad (India), Quebec (Canada), and the headquarters in Barneveld (The Netherlands).

In the 90s Baan was considered the largest family software firm in Dutch history<sup>41</sup>. However, by the end of the 90s Baan had run into some financial troubles. Figure 35 shows how the Baan stock price changed between January 1998 and July 2000.

---

<sup>41</sup> The history of Baan Corporation is based on internet sources (Google search for ‘Baan history’) and my personal experience with ERP and Baan Corporation in 1997-1999.

**Figure 35: Baan stock prices (adopted from Baker et al. 2000)**



The main events in the history of Baan Corporation are summarized in Table 11, as described in *Business Week* (Baker et al. 2000).

**Table 11: The Rise & Fall of Baan Co. (adopted from Baker et al. 2000)**

<b>1978</b>	Jan Baan, a high school drop-out and former clerk at a slaughterhouse, founds a software company in his rural hometown of Barneveld, the Netherlands.
<b>1993</b>	Seeing a bright future for enterprise software, Connecticut's General Atlantic Partners invests \$21 million in Baan, buying one-third of the company.
<b>1994</b>	Jan Baan sells the software system to Boeing. The breakthrough contract raises Baan's profile and prepares it for an IPO.
<b>1995</b>	Before Baan lists its shares on Nasdaq and the Amsterdam exchange, the Baan brothers put control of company in the hands of their charitable foundation. In the next three years, the stock soars to 13 times its previous value.
<b>1996</b>	Buoyed by strong stock, Baan goes on a buying spree, snapping up nine different software companies over two years, including Aurum--a rival of Siebel Systems.
<b>1997</b>	With demand for back-office software at an all-time high, Baan revenue soars 91%.

<b>1998</b>	In April, the company's share price peaks at \$54. Then it adjusts first-quarter 1998 sales by \$43 million, explaining that many of the sales were made to its own distribution company. Investors sell down shares 15% in two days.
<b>July '98</b>	The Baan brothers withdraw from the company. Tom Tinsley, a former McKinsey & Co. consultant who joined Baan in 1995, takes over the CEO position. As the Baan stock falls, banks that were holding the Baan brothers' stock as loan collateral unload 8% of the company.
<b>Nov. '98</b>	President Mary Coleman, formerly of Aurum, leads the move to cut 1,200 jobs.
<b>May '99</b>	Tinsley quits, taking a job at General Atlantic Partners, the same VC firm that put Baan on the map. He is replaced by Mary Coleman.
<b>June '99</b>	The Baan brothers' Vanenburg Ventures investment firm, which holds 20% of Baan stock, quietly sells more than half of it by the end of the year.
<b>Jan. '00</b>	With finances plummeting, Mary Coleman quits. New CEO Pierre Everaert searches for a buyer.
<b>May '00</b>	Britain's Invensys announces \$700 million offer for Baan, pricing shares at \$2.85. Aug. 1: Deal goes through.
<b>Late 2000</b>	Two of Vanenburg's new software companies are scheduled for IPOs on the Nasdaq. This includes Top Tier <sup>42</sup> , a key software supplier to longtime Baan rival SAP.

Since 2000 Baan has changed owners twice. In 2000 Baan was acquired for about \$700 million by Invensys (a global automation, controls and process solutions

---

<sup>42</sup> Top Tier was sold to SAP: as a result, some of the teams from Top Tier became part of KM Collaboration group that was the focus of the SAP case study, as described in section 6.1.3.1

group that offers products and services to improve resource productivity<sup>43</sup>). Three years later, in 2003 Invensys sold Baan Corporation for \$135 million to two USA private equity firms.

A recent update to the story of Baan and the Baan brothers is that, after Baan was sold in 2000, Jan Baan started a new company called Cordys, as a part of Vanenburg Ventures. According to the Cordys web site ([www.cordys.com](http://www.cordys.com)), it is developing ‘a Collaborative Real-Time Enterprise Technology platform and ‘beyond ERP’ collaborative Lean Enterprise applications’<sup>44</sup>.

The case study described in this thesis focuses on the development of an E-Enterprise suite that consists of several products. The case study was conducted in early 2002, when two globally distributed locations – Hyderabad (India) and Barneveld (The Netherlands) – were involved in developing software. At that time Baan was part of Invensys. As mentioned in Chapter 4, in June 2002, when I planned to interview people in the Barneveld office, Baan started re-organising its development centres and activities. As a result, in July 2002 development of the E-Enterprise was stopped in The Netherlands, and the Baan facility in Barneveld was closed. Although the Baan case study does not fit the unit of analysis and case selection criteria, including it in this thesis gives an opportunity to compare managerial practices from the successful projects of LeCroy, SAP and TCS with practices that were, or, more importantly, *were not* in place in the unsuccessful project of Baan.

---

<sup>43</sup> From Invensys web site

(<http://www.invensys.com/us/eng/aboutus/whoweare/whoweare.htm>)

<sup>44</sup> Read more about Cordys and Jan Baan in the media release from June 25 2004 on [www.IT-director.com](http://www.IT-director.com)

### 8.1.2 BACKGROUND OF THE PROJECT AND PRODUCT UNDER STUDY

The project investigated in this case study concerns the development of an E-Enterprise suite designed to let users extend their Baan manufacturing, financial, and distribution software on the Web to allow them to collaborate better with customers, suppliers, and partners. According to the *Information Week* of April 26, 1999, Baan then released a first version of the E-Enterprise suite, which included E-Sales, E-Procurement, and E-Collaboration<sup>45</sup>:

*E-Sales* lets users set up an online storefront that Baan says will be integrated with its back-office enterprise resource planning applications. Also included is E-Config, a self-service product configurator that works over the Web.

*E-Procurement* lets companies quickly and easily purchase office supplies and production materials. It also sits on top of the traditional Baan ERP applications and pulls out the operations and business information needed to execute a transaction.

*E-Collaboration* is a lower-cost alternative to electronic data interchange. It lets supply-chain partners share information such as contracts, purchase orders, and material forecasts over the Web. Data generated within the Baan ERP applications, such as a master production schedules or manufacturing diagrams, can be posted on a common site.

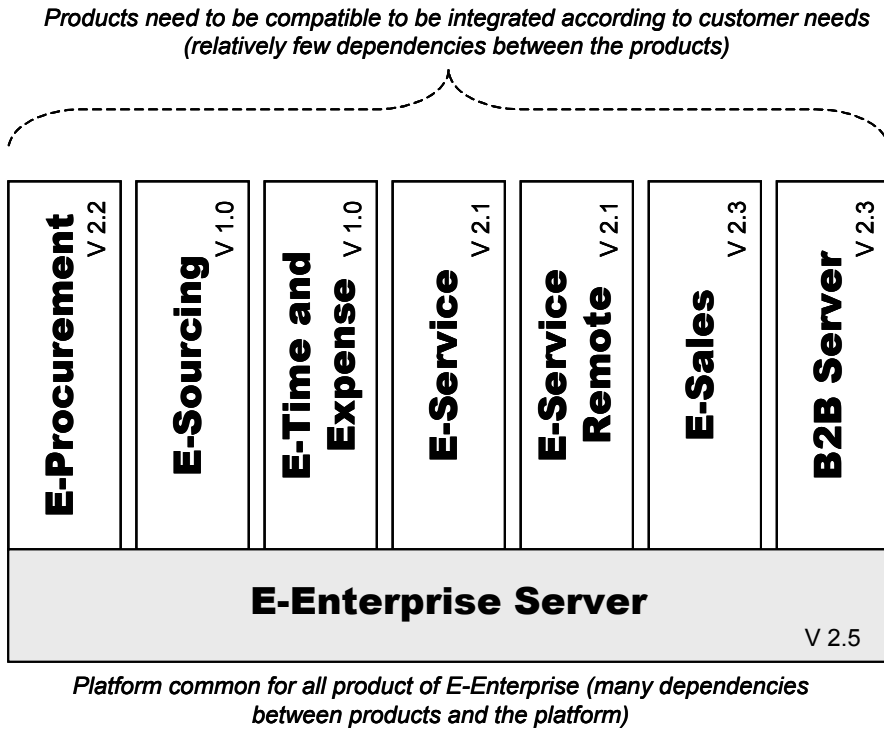
By early 2002, when the data collection took place, the content of the E-Enterprise suite had been extended to include more products. As the interviewees explained, products included in the E-Enterprise suite were developed to be stand-alone as well as to be integrated with the ERP package developed by Baan. In March 2002

---

<sup>45</sup> Extract from the *Information Week* of April 26 1999, available on <http://www.informationweek.com/731/baan.htm>

the E-Enterprise suite consisted of seven products that were all based on one platform called E-Enterprise Server (previously called E-Common)<sup>46</sup>:

**Figure 36: Products included in the E-Enterprise suite**



The E-Enterprise Server included several products that could provide customers with solutions to their business problems. As Venkat Rao (Product Manager E-Service and E-Service Remote) explained:

Customers are not concerned about products but the solutions to their business problems. It could be a combination of products, not only

<sup>46</sup> From the empirical data it seems that the E-Collaboration module was renamed as B2B Server, while in the documents old names are still used (E-Common instead of E-Enterprise Server and E-Collaboration instead of B2B Server)

products but also certain builds, like customisation. Basically the solution is a bundle of products, where products are something like assembling parts.

However, from the development group perspective, products cannot simply be ‘assembled’: ‘products are not so independent. That kind of plug-and-play scenario is not there yet’ (Sridar Bale, Development Manager of E-Time and Expense, E-Service and E-Service Remote).

True Component-Based products can be ‘assembled’ in a plug-and-play manner; however, the *structure of E-Enterprise was not Component-Based*. As Jeevan Reddy (General Manager of E-Enterprise and E-Enterprise India) explained:

In Component-Based development every business function can grow on its own, it need not be dependent on the other functions. Today if I want to grow in one function, it is dependent on the other functions so that I cannot release this function, unless the other functions are also released. Today, if you ask me whether these are components, E-Enterprise is not componentized. Slowly we are moving towards componentization, but we are not there yet.

The software architecture of the E-Enterprise suite was modular: each product included in the E-Enterprise suite was a module, which was dependent on the other modules (dependencies between modules is discussed in Section 8.3).

### **8.1.3 BACKGROUND OF THE SOFTWARE TEAM**

#### **8.1.3.1 WORKING EXPERIENCE IN A GLOBALLY DISTRIBUTED ENVIRONMENT**

E-Enterprise group was relatively young: the first E-Enterprise products were released in 1999. Some people in Hyderabad had been working in a globally distributed environment on other projects: many of them had visited remote locations and worked with some people at a remote site before joining the E-Enterprise group. However, because of a general Baan policy to reduce travel expenses, and because the E-Enterprise organisational structure had changed several times since the group was established (as discussed in detail further, in

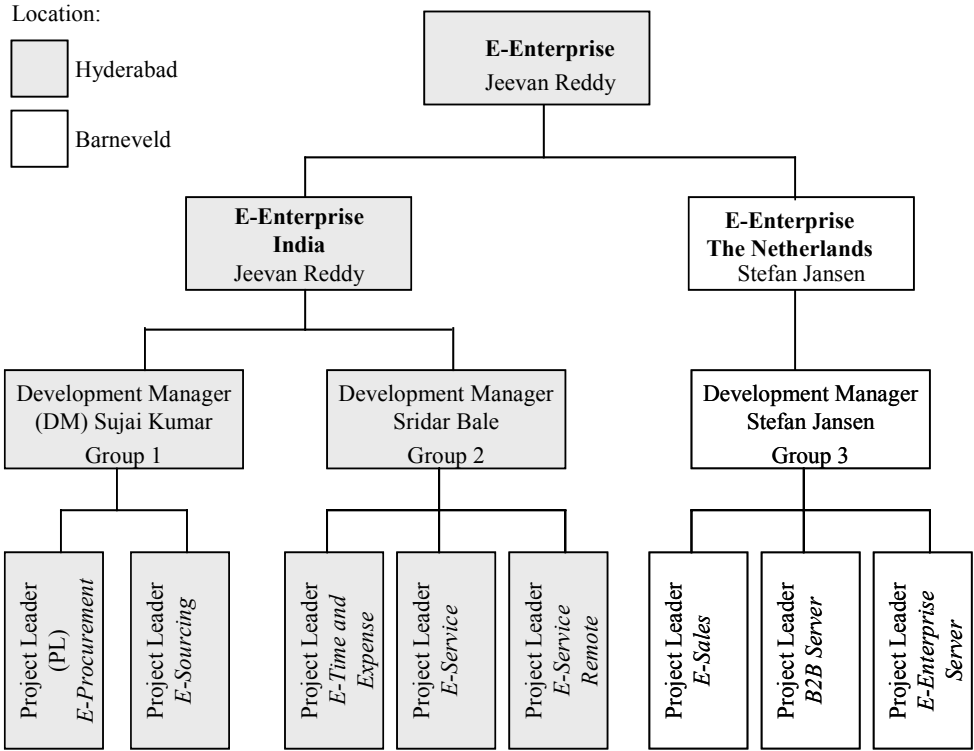


Section 8.3), the vast majority of interviewees in Hyderabad hardly knew people involved in E-Enterprise at the remote site.

**8.1.3.2 ORGANIZATIONAL STRUCTURE OF THE SOFTWARE TEAM**

Development of the E-Enterprise suite was organised by feature/product function (different functions of the E-Enterprise suite are treated as products). A schematic illustration of the organizational structure of the E-Enterprise group is presented in Figure 37.

**Figure 37: Organizational structure of the E-Enterprise development group (as of March 2002)**



From a geographical perspective, the development group was distributed between two locations (numbers are correct as of March 2002):

1) Barneveld (The Netherlands): headquarters with ~35 people involved in E-Enterprise (head of the E-Enterprise group Jeevan Reddy was located in India while the headquarters of Baan, which was also considered the headquarters of E-Enterprise, was in The Netherlands<sup>47</sup>).

2) Hyderabad (India): ~60 people involved in E-Enterprise.

In addition to the E-Enterprise development group, several more groups were involved in the management of the E-Enterprise suite, such as Marketing & Alliances the (M&A) group, and the Project & Process office. In particular, the M&A group had much influence on the E-Enterprise development group. M&A was even considered to be the ‘owner’ of the E-Enterprise and the ultimate ‘customer’ of the development group:

The Product Manager and the Solution Manager [both belong to M&A group], basically are the owners of the product, for us they are end-customers, so whatever they want, we have to do it (Srinivas Ponnada, Product Architect of E-Service Remote).

## **8.2 DATA COLLECTED**

Data was collected from a variety of sources: (i) interviews; (ii) external reports and press releases (I had very limited access to internal project and company documents); (iii) direct observations in the Hyderabad office (I spent ten days there in early March 2002), and the Barneveld office (which I visited briefly twice, the first time in January 2002 before the visit to Hyderabad, and the second time in March 2002 after the visit); (iv) informal conversations with managers and software engineers. Table 12 summarizes the names of interviewees, their roles,

---

<sup>47</sup> The appointment of Jeevan Reddy (who was located in the Hyderabad office) as a head of the E-Enterprise group was an attempt to transfer the development of the E-Enterprise from The Netherlands to India.

locations (Hyderabad or Barneveld team), and details of interviews and other communications for data collection purposes (roles are correct for March 2002).

**Table 12: Baan: Interview and data collection details**

<b>Name</b>	<b>Role and product</b>	<b>Location</b>	<b>Interview details</b>
Sjaak Brinkkemper	Senior process engineer in Project & Process Office	Barneveld	• Initial contact, interview on January 3, 2002
Jeevan Reddy	General manager (GM) of E-Enterprise (and E-Enterprise India)	Hyderabad	• Interview at Hyderabad office on March 11, 2002
Sridhar Bale	Development (DM) manager of Group 2	Hyderabad	• Interview at Hyderabad office on March 12, 2002
Phani Kumar	Product architect (PA) of E-Service	Hyderabad	• Interview at Hyderabad office on March 12, 2002
Sujai Kumar	Development manager of Group 1	Hyderabad	• Interview at Hyderabad office on March 13, 2002
Srinivas Ponnada	Product architect of E-Service Remote	Hyderabad	• Interview at Hyderabad office on March 14, 2002
Venkat Rao	Product manager (PM) of E-Service and E-Service Remote	Hyderabad	• Interview at Hyderabad office on March 15, 2002
P R G Ganesh	Process manager (Hyderabad)	Hyderabad	• Interview at Hyderabad office on March 12, 2002
Vijaya Kumar	Product manager and consultant of E-Time and Expense	Hyderabad	• Interview at Hyderabad office on March 14, 2002
V Maruthi Sivakumar	Product architect E-Procurement	Hyderabad	• Interview at Hyderabad office on March 14, 2002
Johnson Thomas	Product architect of E-Time and Expense	Hyderabad	• Interview at Hyderabad office on March 18, 2002
Sathish Babu	Product architect of E-Sourcing	Hyderabad	• Interview at Hyderabad office on March 18, 2002
Stefan Jansen	Head of E-Enterprise NL and development manager of Group 3	Barneveld	• Interview in Barneveld the end of March 2002

Empirical investigation involved a visit to the Baan office in Hyderabad, and two brief visits to the Barneveld office (the first visit to touch base and plan the visit to India; the second visit to discuss and plan data collection in Barneveld). Data collection was conducted between January 2002 until March 2002, and consisted of the following stages:

- First, initial contact and arrangements for data collection were made with the help of Sjaak Brinkkemper. In January 2002, together with my co-promoter Jos van Hillegersberg, we visited the Baan office in Barneveld and conducted a pilot interview with Sjaak Brinkkemper in order to discuss details of data collection at Baan.
- Then, in March 2002 I visited the Baan office in Hyderabad where I conducted eleven interviews with people in different roles involved in different products comprising E-Enterprise. I spent ten days in the Baan office observing how the software team was working. Furthermore, I attended a Videoconference between India and NL and was present during one conference call that involved members from the M&A and development groups in NL, India and other locations. In Hyderabad I stayed at the same hotel where visiting Baan employees from The Netherlands were staying. This gave me an opportunity to talk informally with Dutch employees visiting Baan Hyderabad and learn their opinion on globally distributed collaboration at Baan in general and within the E-Enterprise group in particular.
- After coming back from India I contacted Stefan Jansen, development manager at the Barneveld office, and visited him to plan interviews with the E-Enterprise group in Barneveld. Interviews were planned for July 2002. However, by that time Baan had started re-organising its development centres and activities. As a result, development of E-Enterprise in Barneveld was stopped, and later the Baan development centre in Barneveld was closed.

### 8.3 HOW BAAN ORGANISES AND MANAGES GDSD: PROBLEMS FACED AND IMPLICATIONS FOR SUCCESS FACTORS

In this section the findings from the E-Enterprise project are analysed and discussed in the light of potential factors contributing to success suggested in the theoretical lens (Figure 15). For each of the four factors, problems identified in the E-Enterprise project are presented, followed by the discussion of critical success factors.

As explained in Section 4.7, the presentation of results of the Baan case is slightly different from the template used for the LeCroy, SAP and TCS cases. The findings from the E-Enterprise project identified several problems faced by the globally distributed E-Enterprise group at Baan, and critical success factors.

Based on these findings I suggest that the problems faced by the globally distributed E-Enterprise group at Baan might have had an influence on the failure of globally distributed development at Baan. It is important to note that these problems were identified *before* globally distributed development at Baan was stopped: the problems are mentioned in the interviews and observation notes that were made in March 2002 while visiting Baan Hyderabad.

#### **(I) Inter-site Coordination: Problems faced**

Interviewees reported a number of problems related to coordination between remote sites. In particular, division of work between the two sites was not efficient: first, ownership of work packages was sequentially switching between teams in The Netherlands and India, which was identified by the interviewees in the Hyderabad office as one of the major problems in the E-Enterprise project (problem 1). Second, too many people were involved in the management of each product in different roles, some of which were overlapping (problem 2). Third, there were many technical dependencies between products included in the E-Enterprise suite, which created knowledge and information dependencies between

the dispersed teams (problem 3). In addition, lack of communications between the two teams caused difficulties in the understanding of dependencies between products and plans.

Furthermore, project management techniques adopted by the E-Enterprise group were not efficient: project planning was too detailed (down to 2-20 hour tasks) and could not accommodate the everyday dynamics (problem 4), which reduced the efficiency of the development and increased bureaucracy, because project leaders were busy nearly full-time updating plans and reports, and developers were busy reporting on the work-hours put into tasks. At the same time, there was a lack of proper planning on a high level. Two problems were associated with lack of proper planning on a high level: first, the requirements stage was not defined (problem 5); and second: there were too many changes in many aspects of the Baan organisation and the E-Enterprise project (problem 6).

Problems 1-6 are discussed below and illustrated by empirical evidence from the interviews.

### **Problem 1: No clear product / project ownership**

Ownership of the project (E-Enterprise) and products comprising it was confusing. Product / project ownership was sequentially switching between India and The Netherlands. Following is the history of the E-Service, illustrating how ownership of the product was switching:

We initiated the project in India in 1999 and developed the initial version. Afterwards we transferred the ownership to The Netherlands because we were busy with an other project, E-Service Remote, where we had some customer requirements which were urgent at that stage, so the entire E-Service team concentrated on the E-Service Remote product. Then the actual ownership of the whole product E-Service was shifted to The Netherlands, and the next version (E-Service 2.0) was developed in the Netherlands: they enhanced the version we developed. Once we delivered the E-Service Remote product, we brought E-Service back to India and we developed a service pack

called E-Service 2.0 SP1 (Service Pack 1). That was one which we delivered last June, now we are working on 2.1 (Phani Kumar).

Because the ownership was switching between the teams, there was always a need to understand the product developed by another team (which is often more difficult than to develop a product from scratch), and there was never a complete knowledge of the product and the logic behind it: for example, as Sujai Kumar explained:

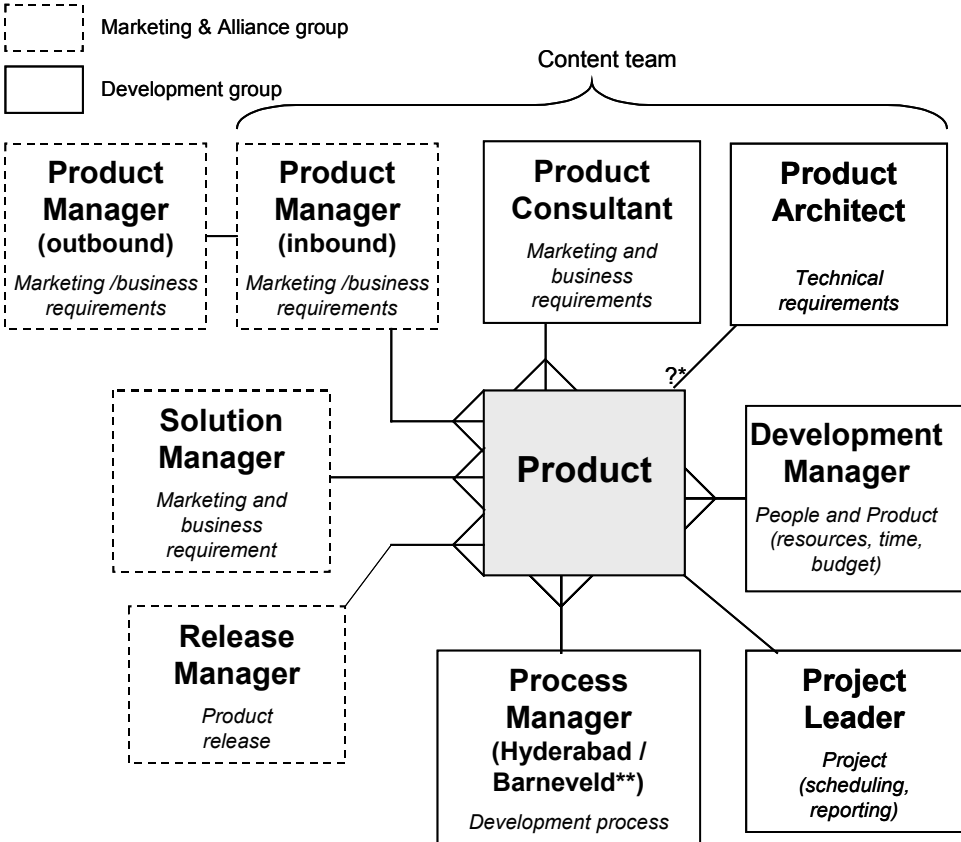
It's difficult to visualise the idea when it is not yours. If we have the knowledge of the existing product then we're building on top of it, it's easy. But sometimes it happens that the understanding of the existing architecture is not very good because we are not there from the beginning: the initial product has been transferred from there to here.

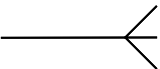
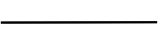
Furthermore, there was no feeling of 'our' product, because the product was inherited from another team: 'I expect one of the important things that should happen within E-Enterprise Baan or anywhere is that more ownership must be felt by everybody' (Vijaya Kumar).

### **Problem 2: Too many people involved in the management of each product in different roles, some of which are overlapping**

It seems that too many people in different roles were involved in the management of each product included in E-Enterprise, so that some responsibilities were overlapping. Combined with other circumstances (e.g. the sequentially changing ownership discussed above), a situation was created where *everybody* was involved but *nobody* was responsible. Figure 38 illustrates the different roles (people) involved in the management of each of the eight products comprising E-Enterprise. Figure 38 is based on descriptions of the different roles as explained by interviewees (the descriptions follow Figure 38). From the descriptions it follows that sometimes people had different views on what they or their colleagues were supposed to do.

**Figure 38: Roles (people) involved in the management of each of the eight products comprising E-Enterprise**



 *One-to-many relationship (one person responsible for many products)*  
 *One-to-one relationship (one person responsible for one product)*

Comments: \* not clear if product architect is responsible for only 1 or for more products  
 \*\* there are two Process Managers – one for Hyderabad and one for NL

There are two *Product Managers*: *in-bound* and *out-bound*. As Venkat Rao (in-bound Product Manager of E-Service and E-Service Remote) explained:

Actually the in-bound and out-bound is more like an internal arrangement. I would say that the in-bound is more product development oriented, whereas out-bound is more product marketing



oriented. Basically, instead of one person taking care of the entire product issues, it is split into two roles: Product Marketer [out-bound PM] and Product Manager [in-bound PM]. The Product Marketer takes care of marketing of the product and also getting the inputs. Getting the inputs is not the primary goal of a Product Marketer, whereas getting the inputs is the primary aim of the Product Manager, and the Product Manager is in the picture of how the product evolves and gets mature as a marketable product. Once it is a marketable product, then from there the Product Marketer takes over.

The in-bound Product Manager is part of a content team.

As Jeevan Reddy (General Manager of E-Enterprise) explained:

*A content team* [see Figure 37] consists of Product Manager, Product Consultant and Product Architect. Each product will have one Product Manager, one Product Consultant and one Product Architect. Some of these people are in The Netherlands: for different products different people sit in different locations. These people are part of the team here, and similarly part of the team in The Netherlands as well. So this is a generic model.

Within the content team the work and responsibilities are divided in the following way:

The Product Manager gives a product definition in which he gives a brief requirement of what exactly he wants, and the Product Consultant will write a conceptual solution on the requirement: a document outlining what exactly is the business process, and he will explain the requirement in more detail. In the conceptual solution the Architect also will come into the picture and he will explain how this functionality will actually be built into the product, from a technical perspective (Srinivas Ponnada, PA of E-Service Remote).

Srinivas Ponnada explained his role of *Product Architect*:

As an Architect I will be writing the functional and technical designs that include product definition, version definition and conception solution. These two documents are taken as input for the developers. Then the Development Manager starts writing the project plan.

Another Architect, Phani Kumar (of E-Service) said:

when a requirement comes to me I say, ‘this requirement needs this much solution time, this much design time, this much realisation’. Then comes the Project Leader who extrapolates this.

The role of Development Manager seemed to be controversial to some extent. People in different roles had different opinions on what were the responsibilities of the Development Manager, as illustrated by the following quotes:

#### *Opinion of Product Architects*

The Development Manager will be making the project plan. Once the requirements are clear and we have a version definition ready, he will start making the project plan. But since the requirements are changing he has to again change the plans: sometimes it becomes impossible to change the plan every day because requirements change so frequently (Srinivas Ponnada, PA of E-Service Remote).

Furthermore, as Maruthi Sivakumar (PA of E-Procurement) said: ‘The Development Manager has the prime responsibility for delivering the product and he is the one who takes care of all the resource allocation’.

#### *Opinion of Development Manager (himself)*

Sujai Kumar, Development Manager of Group 1, explained his perspective:

My involvement is about the decision-making mostly, to see basically how does this [planning] match with the capacity we have. But on the technical decisions or functional decisions, normally we leave it to functional Consultant as well as the Architect.

#### *Opinion of Product Manager*

The Development Manager is taking care of the schedule of the project, the quality of the project and the people. It was becoming too complex for one person to handle, so in E-Enterprise they created a Project Leader and a Development Manager. There is still a gap there. Still the Development Manager is the one who is supposed to take care of schedule as well as quality (Vijaya Kumar, PM and Consultant of E-Time and Expenditure).

### *Opinion of the General Manager of E-Enterprise*

The Development Manager is responsible for product and people. Overall, the Development Manager still takes the lead in the projects, he is a people-manager. A *Project Leader* is only planning and tracking the progress of the project. And he will work closely with the Development Manager for any resource management, any issues which need to be covered (Jeevan Reddy, General Manager of E-Enterprise and E-Enterprise Hyderabad).

The *Solution Manager*, who is part of the M&A group, is also involved in the management of E-Enterprise. His involvement is associated with some problems, in particular:

We have another person called the Solution Manager, who is actually a boss of the Product Manager, he is sitting in The Netherlands and this causes a lot of problems. The way we see it in our organisation, the way it should work is the Product Manager is representing the marketing team discussions, and here the Product Consultant and the Product Architect will represent the development. So that means that it is assumed that the Solution Manager and the Product Manager work together. But sometimes it does not work, because they are in different locations. So it causes a lot of confusion, because lots of times the Product Manager says 'I will convince him [Solution Manager] to do it the way you are doing it', then later Solution Manager comes into the picture and he says 'no I don't like this' [...] Sometimes we feel it is better to talk to the Solution Manager, because he's the ultimate boss. What happened in one of the products: the Solution Manager himself said 'I want this change', and the Product Manager said 'No I don't think we should do it'. But still the Solution Manager forced everyone to do it (Srinivas Ponnada, PA of E-Service Remote).

There is also a *Release Manager*:

The Release Manager is the person who is part of the M&A organisation. He is basically responsible for releasing the product. He is involved right from the project initiation stage and he'll drive the entire project until it is released to the market. He's the guy who has to be in touch with the sales people, who has to get in touch with the

customers, with Baan development: Project Leaders and Development Managers, with M&A: in-bound and out-bound Product Managers. The Release Managers of different products should interact as well (Jeevan Reddy).

In terms of location, Jeevan Reddy explained: ‘we worked with The Netherlands Release Manager, but now we got one Release Manager here, so now basically the transition is taking place’.

There are two *Process Managers*, one located in The Netherlands and one in India. These two Process Managers work closely together on a process plan to improve the software engineering and development process, guided by the CMMI (Capability Maturity Model Integration) framework. The Process Manager in India is responsible for the group in India, and the Process Manager in The Netherlands works closely with the group in the Barneveld office and is responsible for processes in both locations. As Ganesh, Process Manager in Hyderabad, explained:

We are not specialised in any one area - we are close to the teams and we are intermediaries between specialists and teams. We try to arrange a plan and we try to implement, bring some change, these are the kinds of actions that we do.

### **Problem 3: Dependency of all products on the common platform (E-Enterprise Server) and dependencies between products**

There was a strong dependency between the E-Enterprise Server (earlier versions called E-Common) and everything else, and some dependencies between other products included in the E-Enterprise suite (see Figure 36); these dependencies existed because combinations of products had to work together. For example, as Sridar Bale described:

E-Procurement and E-Sourcing are two applications which use E-Common. Both of them are independent, they can be released to the market, but we need to synchronize them because, if there is a customer who buys both applications, they should work together. So products are dependent because if there is a customer that buys several

products, he wants to see an integrated solution. Then features have to be integrated, or some kind of adjustment has to be made, in such a way that they both work together.

In the first place there were technical dependencies on the E-Enterprise Server. This caused knowledge and information dependencies between the dispersed teams:

***Technical dependencies on the E-Enterprise Server***

Jeevan Reddy explained:

This particular component [E-Enterprise Server], which is also a product, becomes a common or a dependency component for all the products. So you cannot release any product unless the E-Enterprise Server is available. This is a dependency. Because of this product, dependency between Hyderabad and The Netherlands exists.

Technical dependencies on the E-Enterprise Server cause two problems: (1) specifications and (2) schedules across products needed to be synchronized:

When they start working on E-Common, we need to view what is required in this for us [Group 2] at this moment of time. That means we need to already see what is the time-frame, what are the features they are going to incorporate into this particular area (Sridar Bale).

Similarly, Phani Kumar stressed that ‘coordinating requirements [specifications] between different products is a problematic area’.

In order to synchronise schedule and specifications, ‘coordination and a lot of communication is required’, said Phani Kumar; he explained that:

somebody needs to moderate the discussion because everybody independently looks at their product. Collectively we have to find a solution, there comes the sharing of the ways of doing things, so a lot of discussion and co-ordination is required for this. It is expected that the people who are owning E-Common have to be more careful and have to consider all the applications, but in that they are independent.

### ***Knowledge and information dependencies***

The technical dependencies of all products on the E-Enterprise Server created more dependencies and problems, as Vijaya Kumar explained:

The dependency on NL is causing problems. Dependency on information, dependency on knowledge (even in terms of simple design documents, for example, functional designs, or technical designs, they are not complete), dependency on requirements, because everything is centralised in Holland and then that has to be shared with us so that we can proceed. The problem is ultimately extended schedules, they were not able to complete the projects in time.

Taking into account the numerous dependencies discussed above, *there was no structured approach to identify and coordinate these dependencies:*

One thing that is missing right now in E-Enterprise is that at any time you can't look into any document to see what are the exact dependencies involved. Right now they're coming with something like a dependency matrix. But so far we didn't have that. So it's generally like if you want to know tomorrow whatever dependency with another product, you have to actually talk to the team members or the Architect or the Consultant. There is no central store or central repository (Satish Babu, PA of E-Sourcing).

### **Problem 4: Very detailed planning (down to 2-20 hours tasks) and a fast-changing situation do not work together**

As mentioned earlier, the situation at the Baan E-Enterprise group was changing very fast: requirements for products were changing, causing changes in dependencies between products; processes were changing; people and their roles were changing; ownership of products was changing between the teams. At the same time, Baan required very detailed planning: the Project Leaders were busy planning short (2-20 hours) tasks. It seems that Baan put too much effort into planning and controlling whether the work is effective and efficient, so that it became too detailed and not capable of catching up with changes. Thus the effort and resources (man/hours) put into planning, in practice reduced the efficiency of

the development and increased bureaucracy, because the Project Leaders were busy almost full-time updating plans and reports, and developers were busy reporting on the work-hours they put into tasks.

**Problem 5: The requirements stage is not clear and requirements are not frozen; a power game as a management approach to manage requirements**

The requirements stage (in terms of procedure) was not defined:

The requirements stage is not very clear, so it happens that even if the Product Manager is sitting in Hyderabad, even if we sit together and discuss the way we'll be doing our product, at later stage a lot of changes and things come into the picture. Product Manager again gets some new ideas and says what we should do (Srinnivas Ponnada).

Furthermore, requirements were changing continuously, causing (i) difficulties in the planning and management of development for specific products, and (ii) tensions between people involved in marketing, development and across teams developing dependent products.

Srinnivas Ponnada explained the reason why requirements were not frozen:

The marketing takes the lead, so that means whatever the marketing says the development has to do it. Because they say ultimately the marketing team is responsible for selling the product. They will bring the revenues. So that's why development takes a back stage, development has to listen to what they say. It is good if they are really clear what exactly they want, what is the vision, what is the roadmap and things don't change very frequently, but if on their side it is not clear, then it causes a lot of confusion. There is no guarantee the requirements will not change. That's the biggest problem we have.

Vijaya Kumar elaborated on the problem:

For example, if we want to start coding, we need to have a clearly-frozen functional design, and many times that is not possible because it keeps changing. It's a document that keeps improving from one day to another, whenever you have new ideas, new thoughts. And the changes that take place are also not coming through immediately.

It seems to be either the general approach at Baan that requirements cannot be frozen, or the personal approach of the Solution Manager. Srinnivas Ponnada explained:

The Solution Manager gives us a lot of changes, even if the project is going on. He believes that the product should always be open. The requirements cannot be frozen. So we should be in a position to give him whatever requirement he wants. He is that kind of person. Other colleagues [for other E-Enterprise products and Baan ERP] experience the same. The requirements are never frozen.

Interviewees also reported that typically there was a 'power game' between individual people involved (Solution Manager, Product Manager, Architect, Development Manager). It means that the person who is the most powerful (in terms of character or personality) is the one who sets up 'rules of the game': if requirements can be changed, how often and to what extent. This is how Srinnivas Ponnada describes the 'power game':

If requirements are changing depends on who is more powerful. Basically from the organisation point of view it should not happen, but a difference of how much influence he [Solution Manager] can put on the development. If he feels that the development team is not really strong he will request a lot of changes.

For example, Baan Service [ERP group] usually takes the lead role over Solution Manager, he doesn't change a lot of things there, because they [Baan Service group] are more powerful. It is more about the character of the person. The line manager, product architect, they feel that the Solution Manager can't do just whatever he wants. Once he has given requirements, that's it, he cannot change them whenever he wants. The way they do it, once Product Manager or Solution Manager requests for a change, they [Baan Service group] will say, 'OK we will take this', and they will say that it will take long to make the change: 'it will take six months or one year'. So if you start projecting that much time, the Solution Manager will never come back with a change, because he knows that they will always say another thing.



### **Problem 6: Too many changes in many aspects of organization and project**

The E-Enterprise project and the Baan organization were continuously changing: people, their roles, products, product's requirements, processes, ownership and physical location of tasks – all was changing very fast. Everything seemed to be in a transition and unstable. This situation reduced morale in the organization and increased tensions between Indian and Dutch group members. Every interviewee mentioned several aspects that had changed recently, for example:

- *Change in product: from E-Common to E-Enterprise Server*: 'there was always migration from E-Common to E-Enterprise Server going on. So when we started with our project, that is E-Source, we started with E-Common, then E-Enterprise Server took over from E-Common' (Satish Babu)
- *Changing ownership: moving tasks between India and The Netherlands*': as described earlier (problem 1)
- *Organizational structure was changing*: for example, the role of the solution manager had been changed: 'The actual solution manager for the product manager has moved out. Previously we had a concept for solutions, so they said for the time being we will remove that concept' (Srinivas Ponnada).

It seems that there was no clarity about the changes within Baan: for example, another interviewee (Venkat Rao) defined the new role of the former solution manager differently:

Actually the solution manager has been re-presented as group manager now. What happened was, there was some confusion about the term 'solution', so they dropped the term 'solution', but still you have a name called group manager. But the group manager, if you want to look at it practically, it is nothing but a solution manager.

This gave the impression that people were used to changes (and expected more changes) in the organisational structure: 'organisational structure is slightly

different now, in the sense that today it's like that, tomorrow our organisational structure can change' (Sridar Bale).

- *Processes were changing:* 'earlier we had BDM (Baan Development Method), now we have D-method. It advocates certain ways of finding requirements, we also have been advised to do that kind of finding of requirements' (Venkat Rao).

Problems 1-6 discussed above can emerge in co-located as well as in globally distributed software development projects. However, these problems become more critical and more difficult to solve in a globally distributed environment where teams cannot meet face-to-face but need to collaborate and solve problems over distance.

### **(I) Inter-site Coordination: Implications for success factors**

Three main critical success factors related to inter-site coordination were mentioned by interviewees: (i) communications between key people, (ii) clearly defined ownership, and (iii) a centralised plan that includes all dependencies. The following quotations illustrate the importance of each of these factors.

#### ***Communications between key people***

- One critical success factor is communication between people who are supposed to be involved very closely in the development: communications between Product Manager and Development Manager, communications between Product Consultant and the development teams, communication between the Consultants because there are so many dependencies between products (Jeevan Reddy).
- The bottleneck that I see here is communication and understanding. These two are very important. Definitely a visit to the other country is going to give a lot of added value in understanding people. Personal understanding, definitely, and building up personal relations (Ganesh).

However, visiting the other country was difficult for people involved in the E-Enterprise group because Baan was ‘trying to make cost-cutting measures, and they tried to shift everything to one location to reduce the communication costs’ (Sridar Bale).

***Ownership clearly defined***

- Involvement of the people and clear ownership is important. It has impact on people, their commitment and motivation. If people working on the product do not feel this ownership, then they are not motivated and not committed. They do not get involved. They do not understand the various dependencies and they'll not work towards the target goal (Vijaya Kumar).

***One centralised plan with a clear requirements matrix and a dependency matrix for coordination and control***

- Based on a successful project related to Baan ERP in which Vijaya Kumar was involved before he started working on E-Enterprise, he explained:

Initially the broad rule is that there will be one plan, not two plans, and the plan has the requirements matrix clearly defined. We know that there are 170 requirements to be done: for each requirement there is a spreadsheet made especially for it, which area it goes in, and who is the owner, who is the consultant, and who is the technical owner for this development matrix. The requirements matrix clearly defines where all the ownership lies and who is the contact. For a period of time that has become the key factor for controlling the whole thing.

**(II) Appropriate Tools and Technologies: Problems faced**

The E-Enterprise group was well equipped with tools and technologies required to enable working in a globally distributed environment. There was only one problem reported in regard to tools and technologies: lack of configuration management tools and methods. In particular, there was lack of compatibility between versions

of different products (problem 7). This problem is discussed below and illustrated by empirical evidence from interviews.

### **Problem 7: Lack of compatibility between versions of different products**

It is important to ‘ensure that whenever a product is changed and a new version is released, it should be backward compatible’ (Satish Baby). Backward compatibility means that the new product (version) should recognise and work with all previous versions of other products of the E-Enterprise suite (the same as a new version on MS Word would recognise Word files created in earlier versions of Word). Satish Baby gave an example:

If today we go into the market with E-Source 1.0 version. Then, during some time B2B Server would have released 3 versions. Now the customer should be in such a position that with E-Source 1.0 he should be able to buy any of these three versions of B2B Server.

However, in practice products included in E-Enterprise were not backward compatible, and this created additional dependencies between products because only specific versions of specific products could work together. Therefore, ‘for each product we need to know specific properties, for example, in a product scenario [combination of products] which versions are in, which release works with which version of E-Common’ (Sridar Bale). As versions of products were not backward compatible, compatibility was managed manually by creating lists of compatible product versions, for example by listing versions of E-Enterprise Server compatible with other products (see the extract of the internal document in Figure 39, and the full document in Appendix 5) and by documenting connectivity packs (see full internal document in Appendix 6).

**Figure 39: Compatibility between versions of different products (extract from Baan product compatibility matrix)**

**Applications running with which E-Common version 1)**

E-Sales 2.0	E-Common 2.0
E-Collaboration 2.0	E-Common 2.0
E-Procurement 2.0	E-Common 2.1
E-Sales 2.1	E-Common 2.2
E-Collaboration 2.1	E-Common 2.2
E-Procurement 2.1	E-Common 2.3.0
E-Service 2.0	E-Common 2.3.1
E-Service Remote 2.0	E-Common 2.3.1
E-Sales 2.2 LA	E-Common 2.3.1
E-Sales 2.2 GA	E-Common 2.3.1 SP1
E-Sales 2.2 SP1	E-Common 2.3.1 SP2
E-Collaboration 2.2	E-Common 2.3.1 SP2
E-Procurement 2.1 SP1	E-Common 2.3.1 SP3

1) Applications which are running on the same E-Common version are able to run together on one server.  
 2) Applications which are running on E-Common 2.2 or higher are able to run together on one server

Management of multiple versions of different products without proper configuration management tools is difficult even in a co-located environment. In a globally distributed environment management of multiple versions manually would require seamless coordination between dispersed sites and complete awareness of what is happening at the remote site. Otherwise it is impossible to manage multiple versions manually across dispersed locations (as happened in the E-Enterprise project).

**(II) Appropriate Tools and Technologies: Implications for success factors**

Tools and technologies were considered very important: ‘this is actually one of the most important things: technology comes to our rescue in working in a distributed environment’ (Venkat Rao). Different tools were used to save on travel costs between The Netherlands and India, as Venkat Rao explained:

Quite some time back, before all of these tools came into practice, we used to travel to The Netherlands and they used to travel here in order to meet us, especially at the start of a new release or to share some important needs that stretch over a long time. Even for small purposes

people used to travel. That was becoming expensive and they [Baan] had to think of alternatives, then all of these media came in the picture. Then the Videoconference was immediately applied. We started using VC, and we don't have to go to The Netherlands: we are saving a lot of dollars.

Interviewees mentioned several attributes of software development tools that are important for working in a globally distributed environment. Furthermore, I asked them what collaborative technologies they use and how they choose media for different purposes. Software development and collaborative tools used in E-Enterprise are described below:

***Software Development tools***

In order to support GSD, interviewees identified the following capabilities that need to be supported by SD tools (described in Table 13).

**Table 13: Capabilities of SD tools at Baan**

<b>Standardization of tools and methods across locations</b>	<p>Baan tried to standardise development methods and processes:</p> <p>We want to have common processes across the locations. We try to achieve a uniform standard for all these. So that is a basic aim of this. Though we have not reached it in all the areas, but in certain areas we are making steps (Jeevan Reddy).</p>
<b>Centralisation of tools</b>	<p>There was an attempt to have a central requirements database; however requirements were changing so quickly that the database was not up to date.</p>
<b>Synchronization of code</b>	<p>Code was synchronised via synchronisation of databases at two locations. Maruthi Sivakumar explained how it works:</p> <p>Generally we have what we call ‘sources’ [source code], we have other sources that are shared. For modules that we have ownership of, whatever files</p>

---

are modified under this particular module they are the sources that are present in the Indian server. They are the leading sources. Then we have a synchronisation mechanism wherein we synchronise both the databases at the same time.

---

### ***Collaborative technology***

The following are collaborative technologies that were used by the E-Enterprise team to collaborate over distance (described in Table 14).

**Table 14: Collaborative technologies used in Baan**

---

**Online chat**            Hyderabad group could use AOL for chat. However, it seemed that chat was used very seldom for communications between The Netherlands and India, if at all (only one interviewee mentioned the existence of chat).

---

**Phone and teleconferencing**    As interviewees explained, the phone was used in the following situations:

Telephone usually involved when a lot of emails have exchanged and certainly we feel that everyone is talking differently and it is taking too much time and no one is coming to any conclusions, then we start organising a telephone call (Srinivas Ponnada).

Furthermore, ‘sometimes when the issue is very urgent and you need to get a reply very fast, then also we use phone’ (Maruthi Sivakumar)

Phani Kumar explained:

If it is complicated or I feel mailing would really be inadequate at that stage, then what we do is we simply call them.

The attitude of some interviewed towards the use of phone can be described as ‘we try to minimise the way we have to talk over the telephone as far as possible. One reason is it being expensive’ (Phani Kumar).

---

---

**Email**

There were different opinions about the use of email, in particular regarding preferences between email and phone.

For example, Phani Kumar prefers to use emails:

If I require some quick queries I generally use the mail, because there is no point in phoning them up. But when there is an issue, then I would actually prefer to send a mail even in that case, because the other person is not aware of the full background, so I try to prepare a document for detailing some of these concerns. If we are unable to sort out the issue via mail, we try to have a conference call. By mails you can express things more clearly because when you are on a telephone you can't just go on elaborating the things which you want to solve, but explain in a mail so that the other person has time to read, contemplate and then prepare his responses. So better have a telephone conference only at that point and with a fixed agenda.

Satish Baby has different opinion:

Telephone clears lots of things much better than if you contact by mail. Mail I think is not the right medium for high-level discussing requirements or something like that, because you are never clear what the other person understands.

---

**Application Sharing**

Net Meeting and Webex (a Web-based conferencing tool) were often used for meetings between sites and with customers. In particular Webex was convenient:

If you want to talk to a customer, for example, if you want to give a knowledge transfer in something like 1-2 hours, I call a Webex meeting then ask all the parties to log in to that meeting at a given point of time. As a chairman, once you start the meeting and you see people logging in, you can use the telephone for conferencing. Then you start sharing the application or you start sharing the presentation (Venkat Rao).

---



---

**Videoconference** On the one hand, ‘videoconferences are fairly heavy in equipment, heavy in the sense that it uses a lot of performance. It needs a fairly big network. You can use a videoconference from point to point’ (Venkat Rao).

On the other hand, videoconference was considered important because it allowed people to see each other and see each other’s emotions:

To bring everybody in synch we had many people participating in a telephone conference. But then we realised that we were not able to see each other's emotions, we were taking decisions and sometimes arguments used to be a little bit heated. Heated in the sense that sometimes I don't agree with what they say and vice versa. We were getting too emotional, and it also became a bit of a fight. Then we realised why not use the VC?! We have a centralised videoconference room, one in India, one in Holland. We decided to stop Net Meeting and go for VC. Then we fixed up a lot of videoconferencing, twice to three times a week almost (Vijaya Kumar, based on his experience in a successful Baan ERP project).

---

### **(III) Social Ties in GSDS: Problems faced**

Interviewees reported a number of problems related to social and human aspects. In particular, there was a lack of team atmosphere between teams in Hyderabad and Barneveld: from interviewing members of both teams, tensions between the teams became evident, and teams were not motivated to work together (problem 8). Furthermore, many of people interviewed did not know in person their remote counterparts: Baan tried to reduce project costs by reducing travel costs, thus reducing the opportunity of remote team members to meet in person.

Problem 8 is discussed below and illustrated by empirical evidence from interviews.

### **Problem 8: Tensions between Indian and Dutch groups**

I observed and was told about tensions between the Indian and Dutch groups. The following quotes show the tension existing between the groups:

- When *we* gained a lot of knowledge (for example myself: being consultant, I knew the product in and out), *we* realised that *we in India* could take the ownership of the entire product, one module at least, and create everything from scratch. So then *we* really had a huge problem with *Holland* to take ownership. *We* wanted to build a product in India without any influence from Holland, but *they* were not willing to give (Vijaya Kumar).
- The major issue is that people don't perceive that on the other side, *they're* not reciprocating our needs: what *we* want, during which time, what priority *we* have. *They* don't see the same priority as *our* people see, and vice versa. So there is always a gap (Jeevan Reddy).

This problem is not unique to GDSD projects: each nation has its own unique characteristics (Hofstede 1993) that may lead to misunderstandings and conflicts between people with different cultural backgrounds involved in a GDSD project, as it happened in the E-Enterprise project.

### **(III) Social Ties in GDSD: Implications for success factors**

To learn about the importance of social ties in globally distributed software development I asked interviewees if it was important for them to know personally their remote counterparts; and if so, what had changed after they met face-to-face. All interviewees considered that knowing personally and building relationships with remote counterparts was very important for success. Following are quotations illustrating the importance of rapport and trust, and the importance of face-to-face interaction for creating rapport and trust.

## ***Importance of rapport***

*Rapport may reduce the need to travel in the future*

- Talking about his colleague in The Netherlands with whom Vijaya Kumar worked earlier on Baan ERP:

We have established such rapport that we don't need to visit each other anymore. Whatever he says I understand, whatever I say he understands. Even when you send a mail, the meaning of the mail, the way the sentences are formed and the meaning out of it is extremely easy to gather (Vijaya Kumar).

*Good relationships between individuals may reduce problems between remote sites*

- If the marketing people, Solution Manager or the Product Manager, are on good terms with the development team, the Product Architect and the Consultants, things will go on smoothly. We don't need any process and any rules. But if they're not on good terms, like if a lot of things are changing every time from the Solution Manager, Product Manager, then definitely it will be all this kind of problems. Things are managed built on relations (Srinivas Ponnada).

*Knowing in person improves understanding between remote counterparts*

- I think it really helps knowing this particular person, because you know how the person reacts, and when you're expressing or explaining the things (Sridar Bale).
- One thing I've realised in software over this period of time is that there's no one single way of doing things. So when you want to discuss and then come to a conclusion [agreement] on something, you need to understand various things about the person's intent. When you have rapport established, understanding on a personal level, you will also be able to appreciate and reason out why we took this decision and why not that decision. That becomes a very important thing I think (Vijaya Kumar).

## ***Importance of trust***

*Trust (confidence, mutual respect) makes it easier to collaborate over distance (easier to approach somebody, easier to understand, easier to reach an agreement)*

- Asking interviewees at Baan about the importance of knowing their remote colleagues personally, I was told that:

It [knowing remote colleagues personally] builds the confidence. Confidence in the sense that now I can depend on him, because now we understand each other. Even if I go to him, then whether it is right or wrong, he will give me advice or he will give his opinion. I'm building a confidence in me to go to him. So some kind of a mutual respect comes. Then there is a higher transaction, then you can further collaborate much more easily (Jeevan Reddy).

- Talking about his former counterpart in The Netherlands with whom Vijaya Kumar worked in a successful Baan ERP project:

I got to know more about the person and about the value-system that he has, then I realised what kind of person he is. He can be uncomfortable (he can straightaway say that what you say is absolutely wrong and not acceptable), a little bit harsh, straightforward and direct, but then this is in his nature, that's what I realised. So after that experience it was so good and so pleasant to interact with him, and it just went off so smoothly.

- Regarding relationships with remote colleagues, I was told:

If your personal relationship is not good, the issue will either die down or it will be just casually taken. If the relation is good, you have mutual interactions, then he might go out of his way. If the confidence and trust are built, he'll stretch himself to a greater extent. That may not happen if that is not there (Jeevan Reddy).

### ***Importance of face-to-face meetings***

*Meeting in person improves understanding between remote counterparts, makes it easier to communicate, as illustrated by the following quotes:*

- After going through face-to-face discussions and started understanding each other I could see a lot of change in the way we deal with things. Issues are still issues, but now the issues are tackled differently. How is he responding to my need and how I am responding to his needs. There is a change. During face-to-face we shared with each other what are the issues and discussed each other's wishes. So some kind of empathizing is coming in. Understanding each other. To some extent yes, it helps (Jeevan Reddy).
- Personally I feel meeting the people would help you resolve the tasks more quickly, because you can really think and feel the person when you are actually talking. For example, assume two people, one has never come to India and the other has never gone to Holland. If they are interacting, there would be some gaps. But if they had an interaction at a personal level at some point in time, then the interaction would really be better, the response will be generally quicker (Phani Kumar).
- Until you have face-to-face relation, it's very difficult to really judge how that other person is and what techniques I can use for convincing. So it's really important that I know you personally and you know me personally. That is my strong feeling on that. Then you will be successful and we will have an effective communication in place (Ganish).

#### **(IV) Knowledge Sharing in GSD: Problems faced**

Despite the fact that the E-Enterprise group had been established several years before this research was conducted, the constantly changing organizational structure and ownership of products (discussed in problem 6) resulted in a situation where the majority of team members at dispersed locations were either new or had moved from another group (e.g. the ERP group that worked on a Baan

ERP product, very different from E-Enterprise). Thus, team members did not have a history of working together: the majority of them did not know each other and did not know the composition of the dispersed team. Therefore, in Baan transactive memory among dispersed team members was not developed.

Furthermore, team members in The Netherlands and India had different cultural backgrounds in terms of national culture (Dutch and Indian), and organisational culture (newcomers and people from Baan ERP group), and did not have a common technical background: there was a gap in common understanding of the technology and the processes team members were supposed to follow (problem 9). Moreover, it was reported that often people in the Hyderabad office were not aware of what was happening in the Barneveld office: they were not updated about changes in requirements and dependencies between the products, and not aware of product and technology roadmaps. Consequently, there was no (or very limited) collective knowledge shared between the two dispersed teams in Baan (problem 10).

Problems 9 and 10 are discussed below and illustrated by empirical evidence from interviews.

**Problem 9: Cultural gaps between people in terms of national culture (Indian vs. Dutch) and organizational culture (Baan culture vs. newcomers)**

People involved in the E-Enterprise experience two types of cultural differences: in national and organisational cultures. Jeevan Reddy expressed his opinion on cultural differences:

In the current scenario there's a lot of gap in the culture. Let me tell you the difference between earlier and now. When we were working in ERP, ERP was understood very well; also the Dutch culture, the Dutch people, because there was continuity in the people, they understood each other very well. But now in E-Enterprise the major difference is because E-Enterprise is a new set of people, even in The Netherlands it's a new set of people. Most of the people have not met

face-to-face, except some key people. It is my perspective, I might be wrong, E-Enterprise overall (both The Netherlands and India) is not part of the ERP culture. Especially in E-Enterprise Hyderabad, you find two sets of people, you clearly see the difference when you start interacting.

Jeevan Reddy explained that people involved in E-Enterprise could be divided into two 'sets of people' who are different in cultural aspects:

1) *People working in Baan for a long time:*

people who have come from an ERP background or worked for 3-4 years on ERP, and moved into E-Enterprise. They appreciate the processes, they understand the issues because they have also gone through them in the past, they also understand how the Dutch culture is.

2) *Newcomers:*

people who have come directly from outside and started working on E-Enterprise products. They have not undergone the process of maturity, they have not understood the Baan culture very well. They are not exposed to the Dutch culture, they are not exposed to the ERP processes. [...] What we found is that it is too much to tell them that they need to follow the process, because the people are just dragged from outside in a multi-national company and, provided need to deal with the Dutch culture, they are not digesting.

**Problem 10: Gaps in understanding products, processes and technology**

There were problems related to the understanding of products (requirements and architecture of the E-Enterprise suite and individual products), of development processes, and of technology that the products are based on. The gaps were caused to a great extent by the combination of two factors. First, products, processes and technology were not established and were changing all the time. Second, even what was decided upon and established was often not communicated, and therefore not known to the remote team in Hyderabad.

***Product: people were not clear about a roadmap for E-Enterprise and individual products***

For example, Johnson Thomas said: ‘in some cases when the product belongs to The Netherlands, we want to know what exactly they are looking ahead: we want to know what exactly they're doing, what is their approach, how they are going about it’.

Srinivas Ponnada said that, according to the M&A group, there is a product roadmap. He gave very strong opinion about this existing roadmap:

It is very vague in terms of what exactly it should contain, this they don't say. Maybe the product roadmap is saying that this year, or this quarter we will be delivering a new product, but it is not clearly specifying exactly what will be the requirements, it is very vague. It [the product roadmap] is there just for the name's sake, just for the profit of it. What it means is that M&A have returned a roadmap just because someone said that they should have it, not because it is their responsibility or it is the result of something.

Srinivas Ponnada suggested that because the vision of the product was not defined, requirements were not clear and were changing all the time.

The vast majority of interviewees said that often product requirements were not clear. One reason that interviewees gave to explain why requirements are not clear was: because for some products a Product Manager who provides the development team with product requirements is at a remote location.

As Srinivas Ponnada explained:

We had a Product Manager who was our boss at that time, he was sitting in The Netherlands and there was a lot of time gap: he will send the requirements, we will try to understand it, there were a lot of email exchanges, telephone calls, what exactly he wants or the way we think. It causes problems - if we are not sitting in one location it is a big problem.

Jeevan Reddy expressed similar opinion:



We started thinking now that Product Managers should be in a place where the development takes place. We find it more logical if these people are here [in Hyderabad], that is from the experience we have seen, because then communication goes very well.

***Technology: new technology was not established, people were not clear about the technology roadmap for E-Enterprise***

As Venkat Rao explained:

Our suite of products in the E-Enterprise is a fast-changing scene. In the case of E-Enterprise, where we work with Microsoft technology right now, probably we move to another technology, but I don't know, it depends on some kind of feasibility study being conducted now. Microsoft itself is changing its platform from time to time: you can see it might be as frequent as 3-4 years. First of all your technological basis is changing, probably changing for better, but we have to adapt to the changing scenarios there. So obviously we can't have a roadmap that would stretch for more than 1-1½ years.

***Processes: gap in common understanding of processes and resistance to following them***

Jeevan Reddy explained:

The processes are not really defined well, so still you find some gaps in having a common understanding on the processes. Slowly, slowly that is getting reduced, but still I can see an issue over that.

Furthermore, there was internal resistance to following the processes, in particular among newcomers:

Whenever we start on a project we will say that these are the processes which we need to follow. But still we find some people are not very keen, they think that 'what advantage do we get if we follow this process?' So some kind of a one-to-one counselling or coaching takes place. It's a slow process. We have to tell them that they have to follow the processes, but even if they follow the processes, the effectiveness will not be there. So I strongly believe the person himself has to be aware, rather than pushed (Jeevan Reddy).

Problems 9 and 10 discussed above are unique to GSDS projects: people from different countries experience differences in national and, sometimes, organisational cultures, and often they have different technical backgrounds. Moreover, breakdowns in coordination between globally distributed teams lead to in gaps in common understanding of products, processes and technology between dispersed team members.

#### **(IV) Knowledge Sharing in GSDS: Implications for success factors**

Interviewees considered that knowledge sharing is important for success, as the following quotes illustrate:

##### *Common knowledge of an architecture / product is required*

- We have an existing architecture and we need to build future products based on this architecture, so understanding the existing architecture is most important in that case, to be able to build on top of it (Sujai Kumar).
- We have completed our realisation from our side and E-Enterprise Server has also completed their realisation. But now we need to integrate these two: E-Enterprise Server to our applications. So for that we need a lot of knowledge of that product, E-Enterprise Server 2.5 (Sujai Kumar)

##### *Common understanding between key people is necessary*

- I think one of the important features in a collaborative framework is the understanding between the key people, the main stakeholders who are architects and consultants and probably lead engineers. If you are working on a distributed ownership, you need that the key software engineers know each other and understand each other. That really helps (Vijaya Kumar).

##### *Common knowledge about culture is needed*

- Common knowledge includes understanding of a culture. For example, Ganesh (Process Manager for Baan Hyderabad) explained that understanding of cultural

differences helps to define better processes that would be acceptable for Dutch and Indian cultures:

When we write the process plan there are a lot of cultural issues that come into the picture. How to deal with this particular area? I can give you an example on quality assurance - a critical area. In the Indian culture, quality assurance is an important topic - people don't mind someone checking the work they do, but if you compare with our counterpart: in The Netherlands sometimes people don't like this. Because the counterpart The Netherlands team have a different culture - individualistic. So there will be some resistance on that front sometimes. Once we understand this and appreciate the cultural factors, then we can define that better plan (Ganesh).

#### 8.4 POSSIBLE IMPACT OF THE ADOPTION OF CBD ON THE SUCCESS OF THE E-ENTERPRISE PROJECT: DISCUSSION

Taking into account that E-Enterprise was not CB, the question arises: *would adoption of CBD help to avoid the problems experienced by the E-Enterprise group?*

In my opinion some of the problems discussed above could have been avoided if E-Enterprise (the products comprising it) had had a CB structure. The possible impact of the adoption of CBD on the success of E-Enterprise project is discussed in Table 15.

**Table 15: Would adoption of CBD help to avoid the problems: discussion**

<b>Problem</b>	<b>Would adoption of CBD help to avoid the problem?</b>
1. No clear product / project ownership	<b>Probably not</b> , because there would still be a need to understand a product developed by another team
2. Too many people involved in management of each product in different roles, some of which are overlapping	<b>Probably not</b> , because management would still be confusing

<p>3. Dependency of all products on the common platform (E-Enterprise Server) and dependencies between products</p>	<p><b>Probably yes</b></p> <p>Technical dependencies between products would be reduced to (standard) interfaces between (business) components (if each product is treated as a business component). This would reduce the problem of synchronising specifications / requirements. However if some functionality is missed out and not included in any product, the problem of lacking component(s) can appear. This means that there would still be a need for synchronisation of requirements but on a conceptual (not very detailed) level.</p> <p>However, CBD also requires careful management of technical dependencies (e.g. as in the LeCroy case), in particular in a globally distributed environment.</p> <p>Knowledge and information dependencies will be reduced.</p>
<p>4. Very detailed planning (2-20 hours tasks) and fast changing situation do not work together</p>	<p><b>Probably yes</b></p> <p>Planning can become simpler if done per business component</p>
<p>5. The requirements stage is not clear and requirements are not frozen</p>	<p><b>Probably not</b>, as long as the requirements stage is not clearly defined. Furthermore, if requirements are not frozen, they would influence the functional requirements for each component</p>
<p>6. Too many changes in many aspect of organization and project</p>	<p><b>Probably not</b>, if organizational structure, people and ownership are still changing.</p>
<p>7. Lack of compatibility between versions of different products</p>	<p><b>YES</b></p> <p>If interfaces are standard and not changing, it would be easier to maintain compatibility between versions of different products.</p>

<p><b>8.</b> Tensions between Indian and Dutch groups</p>	<p><b>Probably yes, to some extent</b></p> <p>To a great extent tensions are caused by dependencies between products developed in different countries (e.g. E-Enterprise Server not being released on time, lack of information and knowledge about dependencies). Thus, reducing these dependencies, possibly, would reduce tensions between the two groups (however it also depends on division of work, e.g. if teams are given full ownership or not).</p>
<p><b>9.</b> Cultural gaps between people in terms of national culture (Indian vs. Dutch) and organizational culture (Baan culture vs. newcomers)</p>	<p><b>Maybe yes, to some extent</b></p> <p>Possibly, adoption of CBD would introduce a new (CB) culture to the organisation. Thus, differences in organizational culture would be reduced as everybody would be at the same level (new) in this new CB culture (as opposed to the current ERP culture vs. non ERP culture).</p>
<p><b>10.</b> Gaps in understanding products, processes and technology</p>	<p><b>Probably yes, to some extent</b></p> <p>CBD methodology includes component technologies and processes. Therefore, adoption of CBD (specific component technology and related processes) would give some clarity to the group regarding processes and technology. However it would not give clarity regarding products that need to be developed, if the marketing team does not clarify it.</p>

It follows from Table 15 that it is likely that the adoption of CBD could have helped to avoid some of the problems discussed above, in particular problems caused by the existence of dependencies between products developed at remote locations: i.e it would have been easier to coordinate and control these dependencies, and tensions between remote groups would have been reduced.

For example, Sujai Kumar, Development Manager of Group 1, explained the difficulties his group was facing because of the current (non-CB) software architecture. He also mentioned the advantages they would have had if the software architecture had been CB:

Originally, when we planned these products [E-Procurement and E-Sourcing], we didn't have a full view of how the products should be and how they have to grow. So as the products have been designed, they are not very componentised, not modularised. When we started adding features, we didn't think about a lot of complexities intervening, so we didn't think about modularizing at that point. Things started growing, the core started growing and it became really huge. Now we feel that we should have narrowed it down. So for both products we have the same issue - modularization.

To improve the product technically we need to change some of the architecture in order to modularize it. Then it will become easy for us to maintain it in the future: instead of modifying a big, huge program, it's easy to handle modules. We could change the modules very easily, and the impact of that component [E-Enterprise Server] on other parts would also be very much lower.

## **8.5 CONCLUSIONS**

In this chapter, the E-Enterprise project was analysed and discussed in the light of potential factors contributing to success suggested in the theoretical lens.

First, the problems faced by the globally distributed E-Enterprise group at Baan were presented and illustrated by empirical evidence from interviews. These problems, reported in March 2002, gave an indication of unsuccessful collaboration in the E-Enterprise project. Based on these findings I suggest that the problems faced by the globally distributed E-Enterprise group at Baan might have had an influence on the failure of Baan to develop software in a globally distributed environment.

Furthermore, critical success factors considered by interviewees as important to make a globally distributed development successful were assessed. Some of these

success factors were mentioned because they were lacking in the E-Enterprise project, and other factors were mentioned based on the experience interviewees had had in other globally distributed projects of Baan that were successful. Interviewees considered four factors suggested in the theoretical lens as contributing to success in GDSD.

In terms of managerial practices, *coordination* between India and Hyderabad was not efficient: first, division of work between the two sites was not efficient; second, there was lack of communications between the team in The Netherlands and India; third, often people in the Hyderabad office were not aware of plans and changes in products and technology originated by the Barneveld office; fourth, project management techniques adopted by the E-Enterprise group were not efficient.

In relation to *appropriate tools and technologies*, Baan tried to standardise development methods and processes across locations. There was an attempt to have a central requirements database: however, requirements were changing so quickly that the database was not up to date. Code was synchronised via synchronisation of databases at two locations.

Regarding *social ties*, Baan did not have managerial practices aiming to build up social ties between dispersed team members. As a result, there was a lack of team atmosphere between teams in Hyderabad and Barneveld. Furthermore, many of the people interviewed did not know in person their remote counterparts: Baan tried to reduce project costs by reducing travel costs, thus reducing the opportunity of remote team members to meet in person.

Concerning *knowledge sharing*, in the E-Enterprise group there was lack of managerial practices aimed to develop transactive memory and extend collective knowledge of dispersed team members. Moreover, there was no managerial practice in place that would aim to educate dispersed team members in new

technologies. As a result, there was a gap in common understanding between dispersed team members of the technology and the processes they were supposed to follow.

In terms of software architecture the E-Enterprise suite was not designed to support reuse. On the contrary, versions of products included in the E-Enterprise suite were not compatible (as described in problem 7), thus creating obstacles to reusing them in new releases.

The E-Enterprise suite and products included in it were developed in an atmosphere of continuous change of technologies and requirements. Development of the E-Enterprise suite at Baan was undertaken without proper investigation of the available technologies and generic products' requirements. As a result, technologies, requirements and interdependencies between the products included in the suite and their versions were constantly changing, thus reducing the efficiency and effectiveness of the ongoing development project. It is possible that adoption of CBD would help to avoid some of the problems experienced by the E-Enterprise group.

The results of this case study show that the interviewees considered four factors suggested in the theoretical lens as contributing to success in GDSD.

It is important to note that out of four potential success factors identified in the theoretical lens, only one – *tools and technologies* – was present at the studied E-Enterprise project. The other three – *inter-site coordination*, *social ties* and *knowledge sharing* – were lacking. This leads to the question: ***is technology alone enough to succeed in a globally distributed environment?*** *Probably not, as we can learn from the unsuccessful Baan E-Enterprise case.*

This chapter presented and discussed the Baan case study. In the next chapter the cross-case analysis comparing all four cases will be presented and discussed.





## **CHAPTER 9      CROSS-CASE ANALYSIS AND RESULTS**

In this chapter the cross-case analysis of the four studied companies will be presented and discussed. First, the similarities and differences between the studied cases are presented (Section 9.1). Then, managerial practices perceived as important for success in GD CBD in the four studied cases are compared, and propositions that suggest relationships between specific managerial practices and dimensions of success are formulated (Section 9.2). Finally, factors perceived by interviewees as contributing to success in the studied companies are compared (Section 9.3).

### **9.1    SIMILARITIES AND DIFFERENCES BETWEEN THE STUDIED CASES**

In many ways the studied projects are similar. Firstly, three out of the four cases (LeCroy, SAP and TCS) comply with the two criteria that guided the case study selection:

- (1) CBD projects are globally distributed between at least two locations of a single organisation;
- (2) The projects are successful (according to the measures of success explained in Section 2.5).

As explained earlier, the fourth, Baan case serves as a counter-case to compare managerial practices identified in the successful cases with managerial practices that were lacking in the unsuccessful Baan case.

Secondly, all four projects satisfy the secondary requirements for case study selection:

- (a) The projects were concerned with new product development. They were interested in long-term collaboration (as opposed to one-time outsourcing projects);
- (b) The overall sizes of the project teams were comparable (25-35 people).

However, there are some differences between the four studied cases, mainly contextual, such as the different countries involved and, consequently, different cultures and different time-zone differences; different types (and granularity) of components; and different histories (number of years) of the remote teams working together. These differences could explain the differences in results across the cases. Table 16 summarises the similarities and differences between the studied cases.

Table 16: Similarities and differences between the studied cases

	<b>LeCroy</b>	<b>SAP</b>	<b>TCS</b>	<b>Baan</b>
<b>Distribution between continents</b> (countries and locations)	<b>Europe-USA</b> Switzerland (Geneva) and USA (NY and Maine)	<b>India-Europe-USA</b> India (Bangalore), Germany (Walldorf) and USA (Palo Alto)	<b>India-Europe-USA</b> India (Gurgaon and Bombay), Switzerland (Zurich) and USA (San Francisco)	<b>India-Europe</b> India (Hyderabad) and The Netherlands (Barneveld)
<b>Cultures</b>	<b>American and Swiss</b> to a great extent: both teams have multiple nationalities	<b>Indian, German and American</b> to a great extent: the team in Palo Alto has multiple nationalities	<b>Indian</b> - all team members are Indian	<b>Indian and Dutch</b>
<b>Time-zone differences</b>	<b>6 hours</b> between Geneva and USA East coast (NY and Maine)	<b>3½-13½ hours</b> 3½-4½ hours between Bangalore and Walldorf (summer-winter) 9 hours between Walldorf and Palo Alto 12½-13½ hours between Bangalore and Palo Alto	<b>3½-13½ hours</b> 3½-4½ hours between India (Gurgaon and Bombay) and Zurich 12½-13½ hours between India and San Francisco	<b>3½-4½ hours</b> between Hyderabad and Barneveld (summer-winter)

	<b>LeCroy</b>	<b>SAP</b>	<b>TCS</b>	<b>Baan</b>
<b>Project duration and Stage of project</b> (at the time when last interviews were taken)	<b>4.5 years</b> <b>Final stage</b> Two weeks before release of the first version of the product	<b>10 months</b> <b>Final stage</b> A few weeks before release of the second version of the product	<b>1½ years (Skandia)</b> <b>8 months (Dresdner)</b> A few weeks before release of the first version of the product (before 'going live')	<b>3+ years</b> <b>Close to completion</b> A few months before release of a new version of (some modules of) the product
<b>History of collaboration between remote teams</b> (before the studied project)	<b>15 years</b> From mid 80s teams in NY and Geneva worked closely together, team members know each other and have long history of collaboration	<b>No history</b> Teams were merged into one group for the project, team members never worked together before	<b>2-3 years</b> Most team members worked together in India on development of Quartz and knew each other before the project started	<b>Very limited history</b> Most team members never worked together before: some of them were new, the others were moved from other groups/projects
<b>Project type</b>	<b>New product development,</b> for large market of potential customers	<b>New product development,</b> for a large market of potential customers	<b>New product development and implementation,</b> for a specific customer	<b>New product development,</b> for a large market of potential customers

	<b>LeCroy</b>	<b>SAP</b>	<b>TCS</b>	<b>Baan</b>
<b>Intended duration of collaboration</b>	<b>Long-term collaboration</b> between teams in Geneva and NY	<b>Long-term collaboration</b> between teams in Walldorf, Bangalore and Palo Alto	<b>Long-term collaboration</b> between people on future implementations of Quartz, team members can be rotated between different projects.	<b>Long-term collaboration</b> between teams in Hyderabad and Barneveld
<b>Type of components in the product (and granularity)</b>	<b>Technical components</b> Fine-grained	<b>Technical components</b> Fine-grained	<b>Business components</b> Large-grained	<b>N/A</b> business modules that are interdependent (as opposed to components that are independent)

## 9.2 MANAGERIAL PRACTICES PERCEIVED AS IMPORTANT FOR SUCCESS: CROSS-CASE RESULTS

In total 22 managerial practices perceived as important in GD CBD were identified in the studied cases. During data analysis these practices were classified into groups that focus on different aspects of the management of GD CBD, in accordance with the four success factors suggested in the theoretical lens (Figure 15). Furthermore, one more success factor, *components management*, emerged from the data.

The majority of the 22 managerial practices were identified in all three successful cases; however, some practices were identified in two out of these cases, and two managerial practices, namely *managing by 'intuition'* and *managing vendors*, were unique to the SAP and TCS cases respectively. Table 17 lists the managerial practices, grouped into the five success factors, and shows for each practice in which cases it was identified (marked as '+').

Furthermore, as explained in the previous section, the Baan case serves as a counter-case that enables us to compare managerial practices identified in the successful cases with the managerial practices that were lacking in the unsuccessful case. Therefore, based on the analysis presented in Chapter 8, the managerial practices that were lacking in the unsuccessful Baan case are marked as '-'; and practices that interviewees from Baan considered as important for successful GDSD but lacking in the studied E-Enterprise project are marked as '-/need' (i.e. lacking but identified as needed).

The use of a counter-case further underlines the significance of the results presented in this thesis, in particular regarding managerial practices that were reported as existing in the successful cases but lacking in the unsuccessful case.

Empty cells in Table 17 are left for practices that were not mentioned in some of the four cases, or when there is not enough evidence whether these practices were in place or lacking in the case. Managerial practice *managing vendors*, which was identified in the TCS case, is not applicable in the other three cases because they did not involve vendors delivering third-party components (thus marked as ‘N/A’ – not applicable).

Following Table 17, the managerial practices listed in the table are discussed, highlighting similarities and differences between the results across the cases.



**Table 17: Managerial practices: comparison of results across cases**

<b>Managerial practices</b>	<b>LeCroy</b>	<b>SAP</b>	<b>TCS</b>	<b>Baan</b>
I) Inter-site coordination				
1 Increasing awareness	+	+	+	-/need
2 Making efficient division of work	+	+	+	-/need
3 Enabling working flexibility	+	+	+	
4 Facilitating tracking of bugs and development tasks	+		+	
5 Enabling flexible PM techniques	+	+	+	-
6 Designing systematic communications	+	+	+	-/need
II) Appropriate tools and technologies				
7 Software Development tools	+	+	+	+
8 ICT infrastructure	+	+	+	+
9 Collaborative technology	+	+	+	+
III) Social ties				
10 Building relationships	+	+	+	-/need
11 Increasing reachability	+		+	
12 Creating and maintaining team atmosphere	+	+	+	-/need
13 Facilitating interactions	+	+	+	-/need
14 Facilitating cross-pollination	+	+		-
IV) Knowledge sharing				
15 Creating transactive memory among team members	+	+	+	-/need
16 Expanding collective knowledge of dispersed team	+	+	+	-/need
17 Managing 'by intuition'		+		
18 Learning new technology	+		+	-/need
V) Components management				
19 Designing for reuse	+		+	-/need
20 Investing in 'advanced development'	+		+	-
21 Facilitating reuse		+	+	-
22 Managing vendors	N/A	N/A	+	N/A

## **(I) Inter-site Coordination in GD CBD: managerial practices**

Following is a comparison across the four cases of managerial practices that deal with inter-site coordination in GD CBD.

### **1. Increasing awareness**

Increasing awareness of *what is going on in the company and the project* was identified in the three successful case studies. It is particularly important for offshore locations, which are Geneva for LeCroy, Bangalore and Palo Alto for SAP, and Gurgaon, San Francisco and Bombay for TCS. As Sudhir Krishna (SAP) explained: ‘Staying here [in Bangalore], often we lose out a lot of information, because people don’t have time to write every small information in a mail and send it across, or they just forget’.

Furthermore, increasing awareness of the *remote team members* was important in SAP because teams in India, USA and Germany had not worked together before, and many of the team members were not familiar with the culture of their counterparts. In LeCroy and TCS increasing awareness of remote team members was less important, because the majority of the remote team members knew each other and had worked together either in a co-located environment (in TCS, while developing Quartz) or over distance (in LeCroy).

Moreover in the three successful cases the importance of increasing awareness of *the environment at a remote site* was identified; it helps to visualise what is happening when a problem occurs and to understand how to solve the problem.

The significance of this practice can be further underlined by the results from the Baan case, where the team members in Hyderabad office reported about lack of awareness of products, processes and technology as one of the main problems in the E-Enterprise project.

Based on the research findings the following proposition can be formulated:

**P1:** *Increasing awareness of what is going on at dispersed locations and about remote team(s) will reduce the possibility of misunderstanding, conflicts and coordination breakdowns.*

This proposition is relevant for CB and also for traditional GSD projects. With regard to CB projects this proposition suggests that, despite the expectations that the adoption of CBD in globally distributed projects will allow remote teams to work more independently, in GD CBD projects the efficiency of dispersed teams is likely to be greater in the teams which are aware of what their remote counterparts are doing, than in the teams that work independently.

## **2. Making efficient division of work**

Interviewees in the three successful projects indicated the importance of efficient division of work for success. The strategies to divide work that managers follow differ somewhat between the studied projects, in particular:

(i) In SAP the *work is divided feature-wise, providing full ownership and responsibility for each team*: i.e. each of the four teams has full responsibility for an entire block of functionality. There are two reasons why SAP gives full ownership to each of the remote teams. First, because the Collaborative tools were developed from scratch: when the project started, teams did not have knowledge about the product. Second, because teams had just merged, they did not have a history of working together. Thus, giving full ownership to each of the remote teams reduced dependencies and therefore, the need for coordination between the teams.

Different from the SAP case, in LeCroy and TCS team members had worked together before, and expertise in different areas of the product was already developed. Therefore, in LeCroy and TCS work is divided according to *where technical or functional expertise is located*. In TCS the expertise is usually at the

main development centre in Gurgaon: thus, a main strategy that TCS follows to divide work is to do maximum work offshore and minimise work onsite.

(ii) In SAP and TCS there is a *division of technical and 'social' responsibilities* that includes establishing reporting channels across the globe.

In SAP and TCS local development managers (in TCS referred to as onsite and offshore managers) are responsible for the division of specific assignments (tasks) between team members and resolving social issues, because they are aware of the local context and circumstances of the team members. In SAP, the second reason to give 'social' responsibilities to local managers is because they belong to the same culture as team members, which makes it easier for them to understand and deal with team members.

Technical responsibilities in SAP and TCS are centralised in the main development centre (Walldorf for SAP and Gurgaon for TCS):

- In SAP design of the overall product architecture and quality of the product are centralized in headquarters (Walldorf): two architects located in Walldorf provide technical supervision to teams in Bangalore and Palo Alto.
- In TCS the offshore team in Gurgaon has technical responsibilities. This team provides technical support for the onsite team that is responsible for implementation of Quartz. This is in line with the main strategy of TCS to do maximum work offshore.

In LeCroy, the managers of the Geneva and NY teams (Anthony Cake and Larry Salant) combine both technical and 'social' responsibilities; they work closely together (and with the architect in Maine) developing and coordinating the overall product architecture between Geneva, NY and Maine. The reason that in LeCroy there is no need to divide technical and 'social' responsibilities might be because Anthony Cake and Larry Salant have worked together for more than 15 years, and each of them visit the remote locations 5-6 times a year. Therefore they can work

closely and coordinate the development successfully without a need to centralize technical responsibilities at one location.

(iii) Furthermore, the importance of *role continuity and ownership of work* was emphasized by interviewees from TCS and SAP. In SAP, for example, teams at dispersed locations have full ownership of a product functionality they are expected to deliver. In TCS, although a project is transferred between onsite and offshore locations, ownership of the work packages stays with the same team: team members are transferred between onsite and offshore locations together with the work packages (components) they are working on. LeCroy also support role continuity and ownership of work packages despite physical location: for example, similarly to the TCS approach, Gilles Ritter continued working on the same functional area of Maui from NY where he was relocated for one year, as he had worked in Geneva.

The importance of role continuity and ownership of work packages can be further underlined by the example of the unsuccessful E-Enterprise project of Baan, where ownership of work packages was continuously changing between teams in The Netherlands and India, which was identified by interviewees from Baan as one of the major problems in the studied project.

The findings of this research lead to the following proposition:

**P2:** *If globally distributed teams have tight relationships and experience of working together, then a skills-based division of work between dispersed team members will be positively related to project outcomes to a greater extent than a division of work by product feature. (Skill-based division of work can be based on technical or functional/domain skills).*

Proposition P2 is unique to GD CBD projects. It will not be relevant for traditional GDS projects, where a skilled-based division of work will create a great deal of dependencies on the source-code level that will need to be managed over distance;

while in GD CBD the dependencies will be limited to interfaces between components and service components.

**P3:** *Changing ownership of a module / component between dispersed teams throughout the project will be negatively related to the product success and to the motivation of dispersed team members to collaborate (work together) in the future.*

According to the proposition P3, the more the ownership of a module / component is shifted between dispersed teams during a product lifecycle, the higher the chances of losing sight of or misunderstanding the original product requirements: (i.e. each switch in the ownership of development is a potential risk for missing out some information and/or misunderstanding product requirements).

In the first place, proposition P3 is relevant in the case of traditional GDSD as it is based on a comparison of findings between the three CB cases and one non-CB case (Baan). It might be that in CB projects changing process or component ownership throughout the project will have less severe impact on success than in traditional GDSD projects. This leads to the following proposition:

**P3a:** *Changing ownership of work packages between dispersed teams throughout the project will be negatively related to the product success and to the motivation of dispersed team members to collaborate to a greater extent in traditional GDSD, than in CB projects.*

### 3. Enabling working flexibility

The interviewees from the LeCroy, SAP and TCS projects suggested that working flexibility, in terms of (i) providing flexible working conditions, e.g. working from home, and (ii) flexible working hours, is important for success. Flexible working hours help to increase the overlap in working hours between locations so that teams can collaborate in real time:

- in LeCroy: early start in NY and late start in Geneva

- in SAP: early start in Walldorf and late start in Bangalore
- in TCS: early start in Zurich and late start in Gurgaon.

Interviewees from Baan did not mention working flexibility as important for success; however, I do not have evidence that would indicate a lack of this practice in Baan.

#### **4 Facilitating tracking of bugs and development tasks**

##### *Tracking development tasks*

- In LeCroy and TCS the need to track development tasks was mentioned for specific, critical tasks that need to be completed quickly (e.g. for tasks on which onsite and offshore teams work around-the-clock by sending them back and forth, as was described in the Dresdner project of TCS).
- In SAP the tracking of development tasks is done within local teams. The overall system functionality is managed by technical architects from Walldorf.

##### *Tracking and tracing of bugs*

- In LeCroy and TCS the tracking and tracing of bugs is particularly important, because for each single bug being reported, several aspects need to be managed, such as:
  - the source of the bug needs to be tracked: it can have originated from one of the customers, or from an internal development team;
  - all components in which code that contains the bug is reused need to be traced, because a bug reported in one product needs to be fixed in all other products that reuse the same code / component.
- Interviewees at SAP did not mention a need for tracking bugs. The interviewees said that bug fixing can be passed from one time-zone to another time-zone, but did not mention the need for specific mechanisms and/or tools for bug tracking.

Interviewees from Baan did not mention the need to track bugs and development tasks as important for success; however, I do not have evidence that would indicate a lack of this practice in Baan.

## 5. Enabling flexible Project Management (PM) techniques

Interviewees from LeCroy, SAP and TCS pointed out that flexible PM techniques are important in large-scale new product development projects, such as the ones investigated in this research. Flexible PM techniques help to accommodate everyday dynamics, and include:

- On a macro level: planning of major project phases (in SAP this included setting up clear objectives for each team; in LeCroy and TCS, teams work jointly on the same objective)
- On a micro level: flexible and not too detailed planning (2-3 week milestones)

The significance of this practice can be further underlined by the evidence from the Baan case, in which very detailed planning (down to 2-20 hour tasks) could not accommodate the everyday dynamics, which reduced the efficiency of the development and increased bureaucracy, because project leaders were busy nearly full-time updating plans and reports, and developers were busy reporting on the work-hours put into tasks.

These findings lead to the following propositions, which are complementary:

**P4a:** *Flexible and not too detailed planning on a micro level by weekly milestones will accommodate everyday dynamics and will allow control to a greater extent than too detailed planning of hourly or daily tasks.*

**P4b:** *Planning of major project phases with clear objectives for each dispersed team will be positively related to success in the delivery of project objectives.*

The propositions P4a and P4b are relevant for CB and also traditional GDSD projects.



## 6. Designing systematic communications

Interviewees from all four companies mentioned the importance of systematic communications for success. However, only in the three successful cases was this practice in place, and in the Baan case this practice was lacking: interviewees from Baan reported problems caused by a lack of communications between key people.

This practice includes organising frequent communications and designing rules aiming to make communications more effective, in particular:

(i) Scheduling systematic and frequent communications, such as regular teleconferences between software managers in dispersed locations; transatlantic videoconferences with all team members every one or two months (mentioned as important by interviewees from all four companies, but lacking in the Baan case).

(ii) Communicating directly to reach an appropriate person. i.e. no hierarchy in communications (mentioned as important by interviewees from the three successful cases).

(iii) Improving the style and content of communications, which helps to reduce the misunderstandings and confusions that typically happen as a result of different cultural backgrounds. Therefore, improving style and content of communications were considered very important in the LeCroy and SAP cases, where people from different cultures collaborate over distance. In TCS it was not important, as all team members are originally from India and have the same cultural background.

The significance of this practice can be further underlined by observations from the Baan case, in which a lack of communications caused difficulties in the understanding of dependencies between products and plans. Interviewees from Baan stressed the importance of this practice for success.

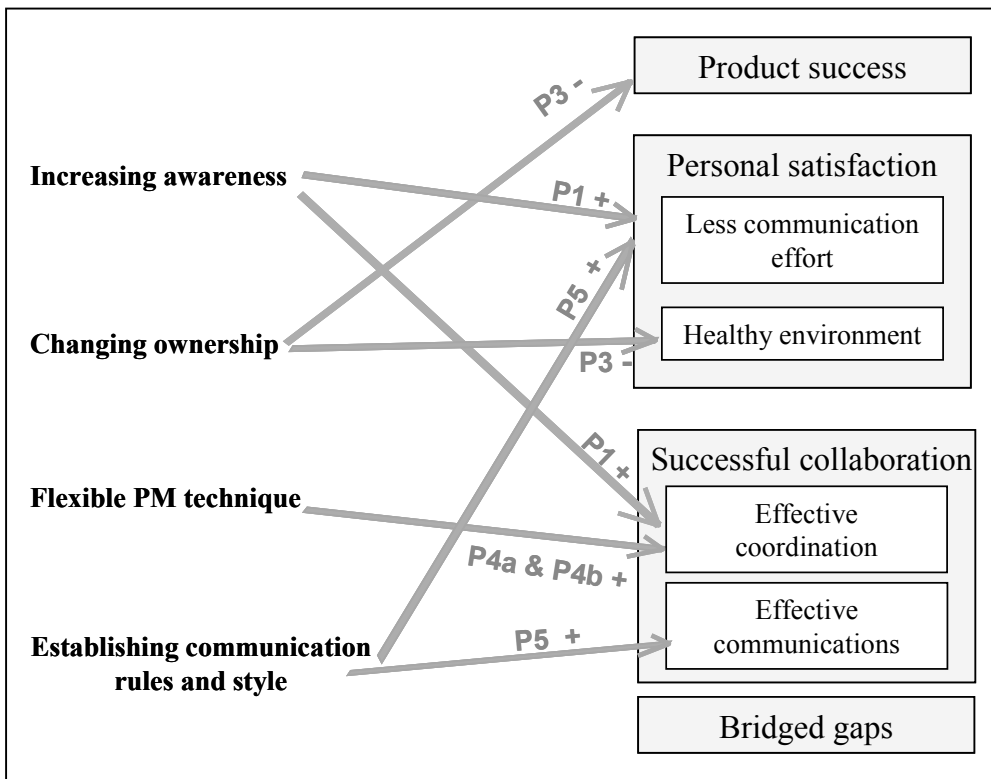
Based on the research findings the following proposition is formulated:

**P5:** *When/If people with different national culture backgrounds collaborate over distance, paying attention to the style and content of communications, agreeing on rules regarding the style and frequency of communications, will reduce the possibility of misunderstandings and conflicts, and will be positively related to the effectiveness of dispersed communications and to personal satisfaction.*

This proposition is relevant for CB and also for traditional GDS D projects.

Figure 40 illustrates the relationships between propositions associated with inter-site coordination and the categories of success (only propositions that connect managerial practices and success in GD CBD are shown in Figure 40. Propositions related to the contextual characteristics of a project are discussed in Section 10.4).

**Figure 40: Inter-site coordination: Propositions**



## (II) Appropriate Tools and Technologies in GD CBD: managerial practices

Following is a comparison across the four cases of managerial practices related to tools and technologies that are important in GD CBD.

### 7. Software Development (SD) tools

**In order to support CBD in a globally distributed environment SD tools need to provide capabilities described in**

Table 18, which compares the results of the four studied projects ('+'- indicated as important, 'N/M' – not mentioned, 'N/A' – not applicable):

**Table 18: Capabilities of SD tools: comparison of results across cases**

<b>Capabilities of SD tools</b>	<b>LeCroy</b>	<b>SAP</b>	<b>TCS</b>	<b>Baan</b>
<b>Automated management of interdependencies between components and related files</b> - supports rapid update of changes by automatically (four times a day) building components that have changed, thus enables the utilisation of time-zone differences (LeCroy).	+	N/M	N/M	N/A
<b>Automated testing of components</b>	+	N/M	+	N/A
<b>Standardization of the tools and methods across locations</b> - using similar tools and methods across locations (LeCroy, SAP, TCS) - replicated development environment of a customer at offshore site (TCS) - in Baan there was an attempt to standardize development methods and processes across locations	+	+	+	+

<b>Centralisation of tools</b> - Web access (LeCroy, SAP) - replicated databases (LeCroy, Baan) - single development environment (LeCroy and SAP) - central repository (LeCroy, TCS) - in Baan there was an attempt to have a central requirements database	+	+	+	+
<b>Creating a Guide that explains how to use tools and methods</b> - documentation about standard tools and methods available on SAPNet (SAP)	+	+	N/M	N/M
<b>Developing tools in-house</b>	+	N/M	+	N/M

Results reported in this research lead to the following proposition:

**P6:** *Standardizations of tools across locations and centralisations of tools in a single development platform/environment will be positively related to greater reuse rate (number of components being reused across different projects/products).*

This proposition is unique to GD CBD.

## 8. ICT infrastructure

Interviewees from the three successful cases suggested that a reliable and high bandwidth ICT infrastructure is required to ensure connectivity between remote sites and make coordination between sites more effective and efficient. For example, Corey Hirsch (VP of IS, LeCroy) outlined:

The role of the WAN, server and applications pool, how file shares are set up, conferencing tools, and just plain network speed are of very high importance [...] and no firm trying to execute GD CBD successfully can do so without the right infrastructure.

Furthermore, appropriate ICT infrastructure needs to support security requirements:

Security is paramount these days, and the internet (plus Microsoft issues) have raised it to a top tier concern. The correct choice of technologies, correct placement of firewalls, correct balance of 'locks', 'police', and 'public awareness' is essential to reduce security risks, while not snuffing out collaboration (Corey Hirsch).

Interviewees from Baan did not mention the importance of the ICT infrastructure, but from my observations (during video- and tele-conferences I attended) I assume that the ICT infrastructure was sufficient to provide appropriate connectivity between dispersed locations. Table 19 illustrates the requirement for ICT infrastructure and compares the results of the four cases ('+'- indicated as important, '-' – lacking, 'N/M' – not mentioned, 'N/A' – not applicable):

**Table 19: Requirement for ICT infrastructure: comparison of results across cases**

<b>Requirements for ICT infrastructure</b>	<b>LeCroy</b>	<b>SAP</b>	<b>TCS</b>	<b>Baan</b>
<b>Quick access to the network</b>	+	+	+	N/M
<b>Shared resources</b> - shared databases (LeCroy) - shared server and project repository (SAP, TCS)	+	+	+	N/M
<b>Web access</b> - constant replication of databases over the Web (LeCroy) - centralised access to tools over the Web (SAP)	+	+	-	N/M
<b>Quick and easy connectivity across locations</b> - use collaborative tools (all four cases) - e.g. dial 5-digit number across the globe (SAP)	+	+	+	+

Based on the results of this research, the following proposition can be formulated:

**P7:** *If the ICT infrastructure provides identical ICT facilities (i.e. similar network speed, server, applications) for teams at dispersed locations as for co-located teams, then the ability of a dispersed team to collaborate effectively and efficiently and the success of project outcomes will be greater, than if the ICT infrastructure provides fewer facilities to dispersed teams than to co-located ones.*

This proposition is relevant for CB and also traditional GDSD projects.

## 9. Collaborative technology

In all four case studies remote team members used collaborative technology extensively. Table 20 illustrates the collaborative technologies that are important for collaboration between remote teams and compares the results of the four cases ('+'- indicated as important, 'N/M' – not mentioned, 'N/U' – not used):

**Table 20: Collaborative technologies: comparison of results across cases**

Collaborative technologies	LeCroy	SAP	TCS	Baan
<p><b>Online chat</b></p> <ul style="list-style-type: none"> <li>- short and/or urgent questions (LeCroy)</li> <li>- in Baan online chat is available for Hyderabad group, but it is not used to communicate with the remote team in The Netherlands</li> <li>- in SAP remote teams do not use online chat; furthermore, need for online chat was not mentioned in SAP</li> </ul>	+	N/U	N/M	N/U
<p><b>Phone and teleconferencing</b></p> <ul style="list-style-type: none"> <li>- urgent matters (all four cases)</li> <li>- update between managers (all four cases)</li> </ul>				

- resolve misunderstandings and conflicts (all four cases) - help in bug fixing (LeCroy)	+	+	+	+
<b>Application Sharing</b> - help in fixing bugs (e.g. show conditions of failure (LeCroy, SAP, TCS) - knowledge sharing (e.g. show slides) (LeCroy, SAP, Baan)	+	+	+	+
<b>Videoconference</b> - progress meetings between managers (LeCroy, SAP, Baan) - major design reviews (SAP)	+	+	N/M	+
<b>Email</b> - low priority tasks (all four cases) - sending source code for small changes (TCS) - sending requirements (Baan) - clarifications (all four cases)	+	+	+	+
<b>Intranet</b> - post internal documents (LeCroy, SAP, TCS)	+	+	+	N/M

Results reported in this research lead to the following propositions:

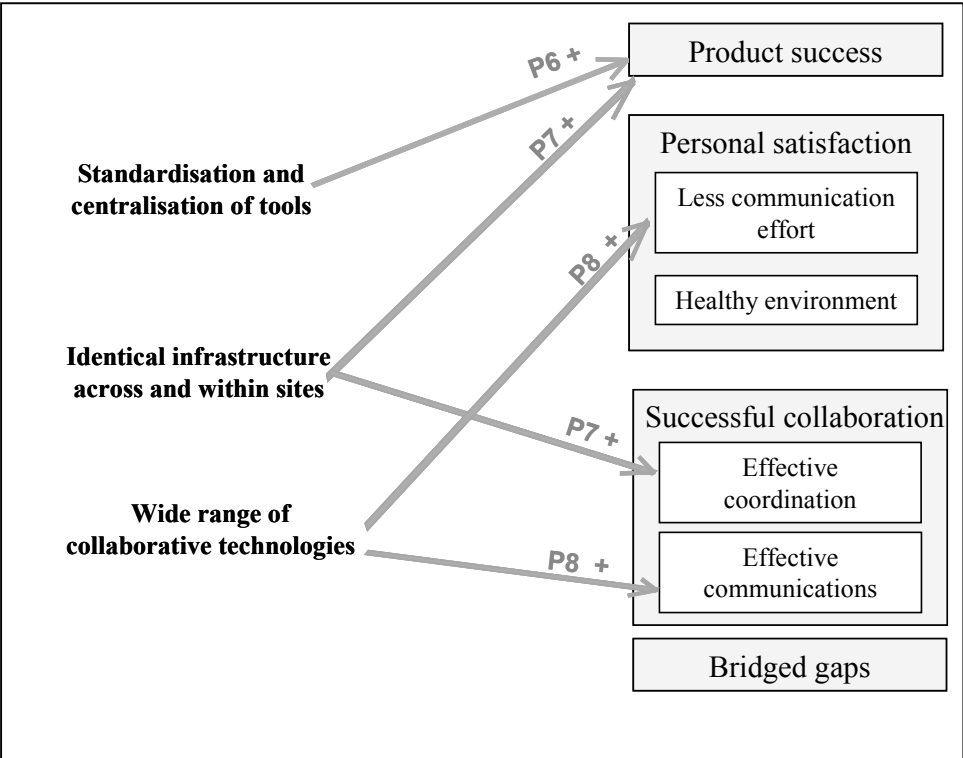
**P8:** *Providing a wide range of collaborative technologies for members of globally dispersed teams is more likely to increase the effectiveness of communications and personal satisfaction than imposing specific types of collaborative technologies to be used.*

**P9:** *Teams/team members who have rapport already developed will use online chat to communicate more often than teams/team members that do not have such rapport.*

These propositions are relevant for CB and also traditional GDSD projects.

Figure 41 illustrates the relationships between propositions associated with tools and technologies and the categories of success.

**Figure 41: Appropriate tools and technologies: Propositions**



**(III) Social Ties in GD CBD: managerial practices**

Interviewees from all companies indicated the contribution of trust and rapport to success. In LeCroy and TCS trust and rapport between remote counterparts were developed to some extent because (i) at LeCroy the software team has a long history of collaborating over distance, and (ii) at TCS the majority of team members have worked together in a co-located environments on the development



of Quartz and knew each other before re-locating to onsite locations (customer sites in Zurich and San Francisco). However, in SAP three teams were merged into one group in the beginning of the studied project, and they had to build trust and rapport from scratch.

In the E-Enterprise project of the Baan the majority of team members in Hyderabad and Barneveld did not know each other; thus, similar to the SAP teams, they had to build up rapport and trust from scratch. However, in contrast to SAP, who invested in building up social ties, in the Baan case the importance of social ties was ignored, which, in turn, led to problems and tensions between teams caused by lack of rapport and trust. These results from the Baan case further underline the significance of social ties in globally distributed teams.

Interviewees from the unsuccessful Baan case emphasized the importance of rapport and trust for success, basing their arguments mainly on their experience in earlier, successful projects, and problems caused by lack of rapport and trust between remote counterparts involved in the E-Enterprise project.

Following is a comparison across the four cases of managerial practices suggested to develop social ties between remote counterparts.

---

## **10. Building relationships**

---

Building relationships involves building rapport and trust between remote team members: it was considered very important for success in all four cases. Interviewees indicated that the best way to build relationships is to meet face-to-face. In LeCroy and SAP, team-building exercises were organised to give developers and key players an opportunity to meet in person in an informal environment. In TCS the majority of team members had met in person on different occasions (e.g. earlier projects, training).

This practice was lacking in the Baan case; however, interviewees from Baan stressed the importance of building relationships between remote counterparts for success, which further emphasizes the significance of this practice.

## 11. Increasing reachability

The importance of increasing reachability was identified by interviewees at LeCroy and TCS: in particular, (i) knowing whom to contact (in LeCroy and TCS), and (ii) knowing who is available on the given day and time (in LeCroy).

Knowing whom to contact is related to the transactive memory of a dispersed team, because when team members know areas of expertise of their remote counterparts, they will know whom to contact.

Interviewees from TCS suggested that because Quartz team members can easily contact each other at any time of the day, they could work faster and utilise time-zone differences to work around-the-clock.

Furthermore, in some cultures, e.g. Indian culture, it is considered normal that one can approach one's counterparts out of working hours, as opposed to many European cultures, e.g. Dutch, Swiss, German, where it is not common to contact somebody about work out of one's working hours. Therefore, the ability to reach somebody out of working hours depends to some extent on the characteristics of a national culture.

Interviewees from SAP and Baan did not mention this managerial practice.

The results of this research lead to the following propositions:

**P10:** *Creating a transactive memory among dispersed team members is positively related to the ability to reach the right people at a remote location.*

**P11:** *Increasing reachability between remote team members is likely to reduce the length of the project.*

**P12:** *The ability to reach the right people at dispersed locations is higher in the cultures with less personal distance or that are more informal (e.g. in collectivist cultures, according to the Hofstede cultural dimensions).*

These propositions are relevant for CB and also traditional GDSD projects.

## **12. Creating and maintaining team atmosphere**

Maintaining team atmosphere was considered important by interviewees from all four companies. In particular, it was important for offshore team members: the team in Geneva (in LeCroy), teams in Bangalore and Palo Alto (in SAP) and the team in Hyderabad (in Baan), because some information / news from the headquarters does not reach remote locations, causing a remote team to feel ‘unplugged’ from the rest of the company, as happened in Baan where the team in Hyderabad felt isolated.

As opposed to LeCroy, SAP and Baan, in each of which major decisions and updated information typically originated from a headquarters office, in TCS there was more balance between onsite and offshore teams in terms of information flows: while offshore teams at the main development centre in Gurgaon were most updated on the technical side of Quartz, the onsite teams (in Zurich for Skandia project and in San Francisco for Dresdner project) had the most updated information regarding customer requirements and progress. In TCS onsite and offshore teams consider themselves as the ‘Quartz family’.

In the Baan case there was a lack of team atmosphere between teams in Hyderabad and Barneveld; furthermore, from interviewing members of both teams, tensions between the teams became evident, and teams were not motivated to work together. This further emphasizes the significance of this practice for success.

This leads to the following proposition:

**P13:** *Creating and maintaining team atmosphere between dispersed teams is positively related to personal satisfaction and motivation to collaborate in the future, and will reduce the possibility of coordination breakdowns and conflicts between the teams.*

This proposition is relevant for CB and also traditional GDSD projects.

### **13. Facilitating interactions**

Facilitating interactions between people at remote locations is considered important in all four cases. This includes (i) facilitating personal face-to-face interactions and (ii) organising regular and frequent interactions over distance.

Interviewees from SAP and LeCroy indicated that personal face-to-face interactions were particularly important in the beginning of a new collaboration, as in the SAP case when several teams were merged into one group, and in the LeCroy case when a team-building exercise was organised in the early stages of the Maui project.

Face-to-face interactions facilitate sharing of knowledge with each other and building relationships. It is an occasion to learn about communication styles; in SAP it was also used to set up rules of communications for future collaboration over distance.

As described earlier, in TCS the majority of team members had an opportunity to meet face-to-face. Therefore, for TCS a major effort in facilitating interactions was put into organising interactions over distance between onsite and offshore project leaders and team members. For example, in the Dresdner project onsite and offshore project leaders had to adjust their working hours to be able to communicate in real time, bridging 13.5 hour time differences.

In the Baan case interviewees stressed the importance of meeting in person and suggested that this improves understanding between remote counterparts and

makes it easier to communicate. However, this practice was lacking in the Baan case, because Baan tried to reduce project costs by reducing travel costs, thus reducing the opportunity of remote team members to meet in person. Lack of this practice in the unsuccessful E-Enterprise project of Baan further underlines the importance of facilitating interaction for success.

These findings lead to the following propositions:

**P14a:** *Facilitating interactions is positively related to building up rapport and trust between dispersed team members.*

**P14b:** *Face-to-face meeting will improve understanding between remote counterparts and increase effectiveness of communications over distance.*

**P14c:** *Rapport and trust (confidence, mutual respect) between remote team members will improve understanding between remote counterparts and increase efficiency of communications over distance.*

These propositions are relevant for CB and also traditional GDSD projects.

#### **14. Facilitating cross-pollination**

Interviewees at LeCroy and SAP considered cross-pollination (i.e. that people from the one group spend a significant amount of time in the remote group and vice versa) to be important for success. In particular, in the SAP case it was helpful in dealing with cultural differences between German and Indian cultures.

Interviewees from TCS did not mention the importance of cross-pollination. This difference between the TCS case and the LeCroy and SAP cases, in which cross-pollination was considered important, can be explained by the fact that all team members of TCS are Indian, thus they do not need to learn about cultural differences.

Cross-pollination was lacking in the unsuccessful Baan case, which further underlines the significance of this practice for success.

Based on the findings of this research the following proposition can be formulated:

**P15:** *Facilitating cross-pollination will reduce the cultural gaps between team members.*

This proposition is relevant for CB and also traditional GDSD projects.

Furthermore, based on all the practices that facilitate development of social ties (i.e. practices 10-15), the following propositions can be suggested:

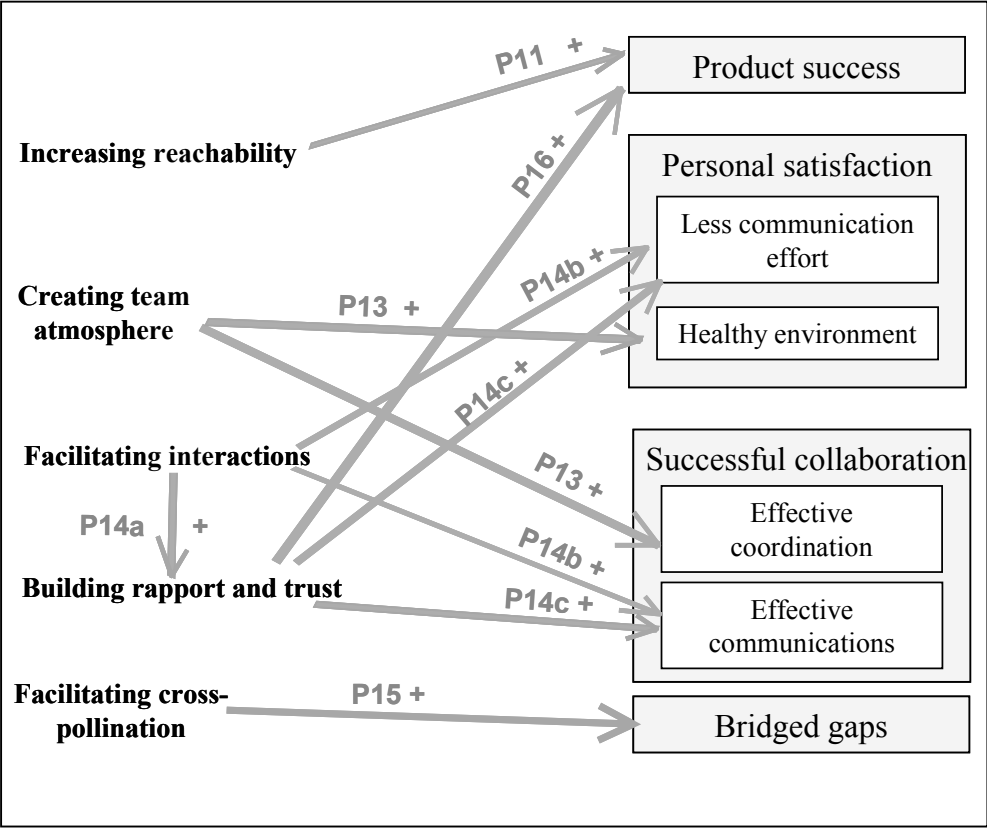
**P16:** *Globally distributed teams in which social ties such as rapport and trust are developed will be more effective and efficient in achieving collaborative project outcomes than teams where social ties are not developed.*

**P17:** *If dispersed teams belong to different national cultures, more efforts by managers, and more investment in terms of time and money, are required to build up rapport and trust.*

These propositions are relevant for CB and also traditional GDSD projects.

Figure 42 illustrates the relationships between the propositions associated with social ties and the categories of success.

**Figure 42: Social ties: Propositions**



**(IV) Knowledge Sharing in GD CBD: managerial practices**

Interviewees from all four cases consider knowledge sharing as contributing to success. Following is a comparison across the four cases of managerial practices suggested to develop transactive memory and collective knowledge in globally distributed teams.

**15. Creating transactive memory among dispersed team members**

Interviewees from all four companies considered creating transactive memory among team members located at remote locations as important for success.

Transactive memory of a globally distributed team implies that team members know the composition of a remote team (who people are, their roles) and know the areas of expertise of their remote counterparts.

In LeCroy, SAP and TCS a number of activities that facilitate interactions among dispersed team members were organised through which team members could get to know each other and create transactive memory. Examples of such activities are: training programs organised in LeCroy and TCS, and team-building exercises organised in SAP; visits to remote locations (in LeCroy and SAP), and rotating people between onsite and offshore teams (in TCS).

Interviewees from LeCroy, SAP and TCS suggested that knowing who knows what at a remote location enables the organisation to reduce development lifecycle because some tasks such as bug fixes can be delegated in an around-the-clock manner (in LeCroy and TCS), and response is quicker when team members know whom to contact for a specific problem (in SAP).

In LeCroy and TCS team members had a history of working together, and many of the dispersed team members had an opportunity to meet in person (the majority in TCS and some in LeCroy), therefore in these two companies transactive memory was developed to some extent (greater in TCS, less in LeCroy) before the case project started, and was also facilitated throughout the project.

In SAP the globally distributed teams in Walldorf, Bangalore and Palo Alto did not have a history of working together before they were merged into the KM Collaboration group. Thus, in this group transactive memory was created from scratch in the early stages of the project through activities such as visits and team-building exercises.

In Baan, despite the fact that the E-Enterprise group had been established several years before this research was conducted, the constantly changing organizational structure and ownership of products resulted in a situation where the majority of



team members at dispersed locations were either new or had moved from another groups. Thus, team members did not have a history of working together; the majority of them did not know each other and did not know the composition of the dispersed team. Therefore, in Baan transactive memory among dispersed team members was not developed. Consequently, the lack of this practice, and the fact that team members in Baan identified the importance of knowing the composition of the dispersed team and areas of expertise of remote counterparts further underlined the significance of this practice.

Based on the results of this research the following proposition can be formulated:

**P18:** *Creating transactive memory among dispersed team members is positively related to collaborative project outcomes (e.g. it will reduce project lifecycle).*

This proposition is relevant for CB and also traditional GDSD projects.

## **16. Expanding collective knowledge of the dispersed team**

Interviewees from all four companies emphasized the importance of collective knowledge shared between members of dispersed teams for success.

Typically collective knowledge is created through shared experiences. In the context of globally distributed teams this means the creating of shared experiences of dispersed team members. Thus, in LeCroy and TCS collective knowledge of dispersed teams had developed before the project started, from the past experience of working together. In particular, in LeCroy and TCS team members also had collective technical knowledge of the Maui platform (for LeCroy) and of Quartz (for TCS). Furthermore, in TCS collective knowledge is very broad, because all team members have the same cultural background (the developers in Zurich and San Francisco are Indian).

However, in SAP dispersed teams (in Walldorf, Bangalore and Palo Alto) did not have a history of working together. Therefore, in this group collective knowledge of these dispersed team had been developing since the start of the project.

In Baan, because of continuous changes in the organizational structure and ownership of products, members of the E-Enterprise group did not have a history of working together. They had different cultural backgrounds in terms of national culture (Dutch and Indian) and organisational culture (newcomers and people from Baan ERP group), and did not have a common technical background. Consequently, there was no collective knowledge shared between the two dispersed teams in Baan. The lack of this practice, and the fact that team members in Baan identified the importance of common understanding between key people, common knowledge of a product architecture and knowledge about the culture of counterparts further underlined the significance of this practice.

Results reported in this research lead to the following propositions:

**P19:** *Expanding collective knowledge of a dispersed (project) team is positively related to collaborative project outcomes (e.g. will reduce a possibility of misunderstandings and conflicts and reduce project lifecycle).*

In particular (propositions P19a and P19b are complementary):

**P19a:** *Expanding common knowledge about national and organizational cultures is (i) positively related to personal satisfaction and effectiveness of communications over distance, and (ii) will reduce the possibility of misunderstandings and conflicts.*

**P19b:** *Expanding collective knowledge related to product architecture and achieving common understanding between key people are likely to reduce project lifecycle.*

These propositions are relevant for CB and also traditional GSD projects.

## 17. Managing 'by intuition'

In SAP the ability to manage 'by intuition' is important for success. Management 'by intuition' is based on catching signals and sensing that something is working or not working properly. To be able to manage 'by intuition' extensive experience in the management of software development in general and globally distributed projects in particular is required.

In LeCroy and TCS this practice has not been reported. However, there is some evidence that implies the possibility that management 'by intuition' is taking place in LeCroy and TCS as well. For example, during my visit to the NY office of LeCroy I observed how software managers Anthony Cake and Larry Salant (who have worked together for more than 15 years) communicate with each other and with software engineers. I observed that they have an intuitive awareness of the situation (environment), of each other and of other people. Similarly, during my visit to the TCS office in Gurgaon, I observed that Sunil Singh and Pankaj Khurana (offshore managers of Dresdner and Skandia, respectively) have intuitive awareness of members of their team and about the situations at remote locations.

The importance of management 'by intuition' was mentioned during my visit to the SAP office in Walldorf, which was the last site I visited for data collection purposes. Therefore, I did not have an opportunity to ask managers at LeCroy and TCS if they relied on intuition, and if so, to what extent (i.e. how important is intuition for managing GD CBD). The need to investigate more in depth the role of management 'by intuition' can be suggested for future research.

Interviewees from Baan did not discuss the importance of management 'by intuition'.

Based on the results of this research the following complementary propositions can be formulated:

**P20a:** *Rapport with remote team members and awareness of what is going on at dispersed locations are positively related to the ability of a manager of a globally distributed team to manage 'by intuition'.*

**P20b:** *The ability of a manager of a globally distributed team to manage 'by intuition' (i.e. catch signals, sense that something is working or not working properly) will reduce the possibility of coordination breakdowns and increase the effectiveness and efficiency of a globally dispersed team.*

These propositions are relevant for CB and also traditional GDSD.

## **18. Learning new technology**

Interviewees from LeCroy and TCS considered learning of a new technology important for success. In both companies this practice included (i) learning a specific component technology used for developing a CB product, and (ii) learning of the CB product, the principles and logic that it is based upon. Learning the design principles and logic was important to make sure that newcomers can understand the product that has been developed already, and will then work following the same principles and logic.

In LeCroy learning new technology involved (i) learning new Microsoft COM technology, and (ii) learning the Maui principles.

In TCS it was concerned with (i) learning the programming language and tools used for developing Quartz, and (ii) learning the theoretical principles and different financial modules included in the Quartz platform.

In both companies intensive courses for learning new technologies were organised.

Interviewees from SAP did not mention this practice. A possible explanation could be that development of Collaborative tools in SAP did not involve the use of a new technology; therefore, team members were familiar with the technology from their experience in previous projects in SAP.

In the Baan case this managerial practice was lacking: as reported in the Baan case, there was a gap in common understanding of the technology and the processes team members were supposed to follow.

The lack of this practice in the unsuccessful E-Enterprise project of Baan further underlines the significance of this practice, which was identified as important in the successful projects of LeCroy and TCS.

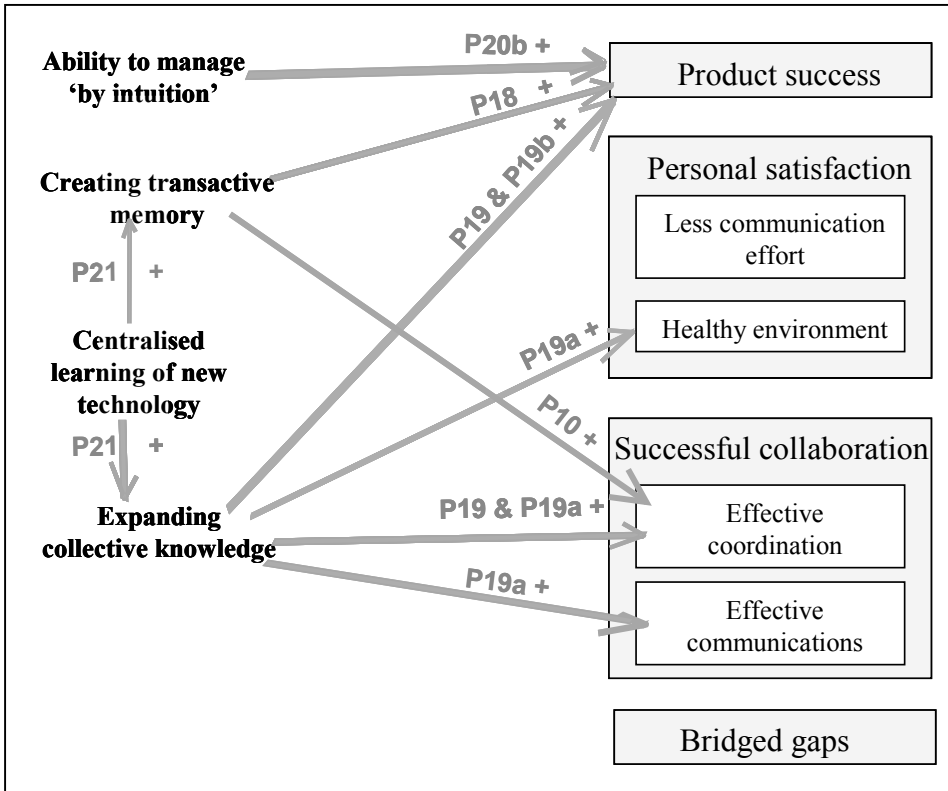
Based on the results reported in this research the following proposition can be formulated:

**P21:** *If globally distributed team members learn new technology in a co-located environment, they will develop more extensive collective knowledge and transactive memory than if training is organised for each dispersed location separately.*

This proposition is relevant for CB and also for traditional GDSD projects.

Figure 43 illustrates the relationships between the propositions associated with knowledge sharing and the categories of success.

**Figure 43: Knowledge sharing: Propositions**



**(V) Components Management in GD CBD: managerial practices**

In addition to the four factors suggested in the theoretical lens, *components management* emerged from the data as a factor contributing to success.

Following is a comparison across the four cases of managerial practices seen as important to ensure successful components management in a GD CBD.

**19. Designing for reuse**

Interviewees in LeCroy and TCS consider that applying a design-for-reuse strategy is important for success.

In both companies (LeCroy and TCS) the main advantage anticipated from developing a CB product was to be able to reuse components in a number of products in the future. Both cases (LeCroy and TCS) aimed to develop a platform that could be extended for a product family (the WaveMaster family at LeCroy and Quartz financial services platform at TCS). In order to maximise reuse across products, software teams of both companies invested time and resources in analysis aimed at identifying the most common functionalities for product families they intended to develop. The analysis addressed the following issues: (i) what components to develop (what functionality is common to all / a majority of products), and (ii) what should be the granularity of the components.

Applying a design-for-reuse strategy in the early stages of a project helped LeCroy and TCS to achieve the benefits of reuse and be more efficient in developing new products based on the Maui platform (for LeCroy) and different implementations of the Quartz platform for different clients (for TCS).

Interviewees from SAP did not mention this practice. However, they mentioned the importance of facilitating reuse (discussed further in this section, managerial practice 21), which emphasizes the importance of facilitating reuse in ongoing projects, while a design-for-reuse strategy emphasizes the importance of reuse during the planning stage of a project, before the actual development has started.

In the Baan case this managerial practice was lacking: the E-Enterprise suite was not designed to support reuse. On the contrary, as described in the Baan case, versions of products included in the E-Enterprise suite were not compatible, which created obstacles to reuse products included in the E-Enterprise suite.

The lack of this practice in the unsuccessful E-Enterprise project of Baan further underlines the significance of this practice, which was identified as important in the successful projects of LeCroy and TCS.

The results reported in this research lead to the following propositions:

**P22:** *In CBD applying a design-for-reuse strategy in the development of a product family will reduce significantly development costs and lifecycle in the long run.*

In particular:

**P22a:** *In CBD applying a design-for-reuse strategy in the development of a product family is likely to increase development costs and lifecycle of a first product and reduce development costs and lifecycle with every new release (of products of the product family).*

These propositions are relevant for globally distributed (and also co-located) CBD projects.

## **20. Investing in ‘advanced development’**

Investing in advanced development was considered important by interviewees of LeCroy and TCS.

In LeCroy, development of the Maui platform was treated not as a typical product development project, when product requirements are defined in the very beginning, but as a research project (referred to by interviewees as ‘advanced development’). It included learning about available technologies, and conducting a feasibility study aiming to test whether or not a ‘proof of concept’ for the product can be achieved by applying available technology(ies).

TCS had a similar approach to the development of the Quartz financial platform. Advanced development in TCS included cooperation with TKS, which was based on integrating core capabilities and knowledge of the two companies – the technical knowledge of developing advanced software products of TCS, and the business knowledge of financial processes, regulations and clients in Europe of TKS.

Interviewees from SAP did not mention this managerial practice.



In the Baan case this managerial practice was lacking: the E-Enterprise suite and products included in it were developed in an atmosphere of continuous change in terms of technologies and requirements. As opposed to the advanced development (i.e. R&D) approach adopted by LeCroy and TCS, development of the E-Enterprise suite at Baan was undertaken without proper investigation of available technologies and generic products' requirements. As a result, technologies, requirements and interdependencies between the products included in the suite and their versions were constantly changing, thus reducing the efficiency and effectiveness of the ongoing development project.

The lack of this practice in the unsuccessful E-Enterprise project of Baan further underlines the significance of this practice, which was identified as important in the successful projects of LeCroy and TCS.

Based on the results of this research the following proposition can be formulated, related to the propositions P22 and P22a:

**P23:** *In CBD approaching the development of a new product as an R&D project is positively related to the ability to reuse components in future products and will reduce development costs and lifecycle in the long run.*

This proposition is relevant for globally distributed (and also co-located) CBD projects.

## **21. Facilitating reuse**

Interviewees from SAP and TCS indicated that facilitating the reuse of knowledge and components across locations is important for success.

In SAP globally dispersed teams organised formal meetings, usually using video-conferencing tools, to discuss what each team had developed and to identify an opportunity to reuse knowledge and/or software components (applications).

In TCS reuse of knowledge is facilitated on two levels: (i) within one project, when people rotate between onsite and offshore to bridge knowledge gaps between the two sites; (ii) across different Quartz implementation projects, which is facilitated by a central person (Quartz program manager) who coordinates all Quartz implementation projects and is aware of new components being developed for a specific customer.

In LeCroy this managerial practice was not mentioned. The reason might be that because in SAP remote teams work on different work packages (each team has full ownership of a work package) and in TCS people from the Quartz group are involved in different Quartz implementation projects, they do not have a direct exposure to the work other teams are engaged in. Therefore in SAP there is a need to have special meetings to learn of what other teams are doing to facilitate reuse across teams; and in TCS a central role (Quartz program manager) is needed to facilitate reuse of components across different implementation projects. However, at LeCroy dispersed teams are exposed to the work of their remote counterparts, first because work is divided based on expertise (skills) and not location, and second because remote teams work in a single development environment (Maui) where they can see what new components have been developed and whether these components can be reused.

In the Baan case this managerial practice was lacking. As mentioned in the ‘designing for reuse’ practice, development of the E-Enterprise did not consider reuse: on the contrary, versions of products included in the E-Enterprise suite were not compatible, thus creating obstacles for reuse in new releases.

The lack of this practice in the unsuccessful E-Enterprise project of Baan further underlines the significance of this practice, which was identified as important in the successful projects of SAP and TCS.

The results of this research lead to the following propositions:

**P24a:** *GD CBD teams that divide work based on skills will achieve higher reuse rate than teams that divide work based on geographical location (i.e. when dispersed teams work on different parts of the project).*

**P24b:** *In GD CBD teams that divide work based on geographical location, if members of dispersed teams do not organise formal meetings to discuss reuse possibilities, it is not likely that they will know about components developed at a remote location that they could reuse.*

These propositions are unique to GD CBD.

## **22. Managing vendors**

Interviewees from TCS stressed the importance of managing vendors providing third-party components: this includes selecting vendors, agreeing on specifications of the components (e.g. functionality and interfaces) and on deadlines for components' delivery.

In particular, vendor management was very important for the Skandia project in which more than 25 vendors were delivering components. In TCS coordination of all dispersed parties, - onsite and offshore teams, and vendors of third-party components - is centralized under the supervision of one person (Quartz program manager) who is responsible for coordinating the work between all parties involved in a Quartz implementation project.

The importance of managing vendors for success was reported in the TCS case only. This practice is not relevant to the SAP and Baan projects as these projects did not use external vendors: all the work was distributed between dispersed teams within one organization. Interviewees from LeCroy mentioned one large vendor in Japan who has been working closely with LeCroy already for several years developing acquisition systems. Possibly because LeCroy is working with fewer vendors than TCS, the interviewees of LeCroy did not stress the importance of

vendor management. Further investigation of this topic can be recommended for future research.

Based on these results, the following complementary propositions can be formulated:

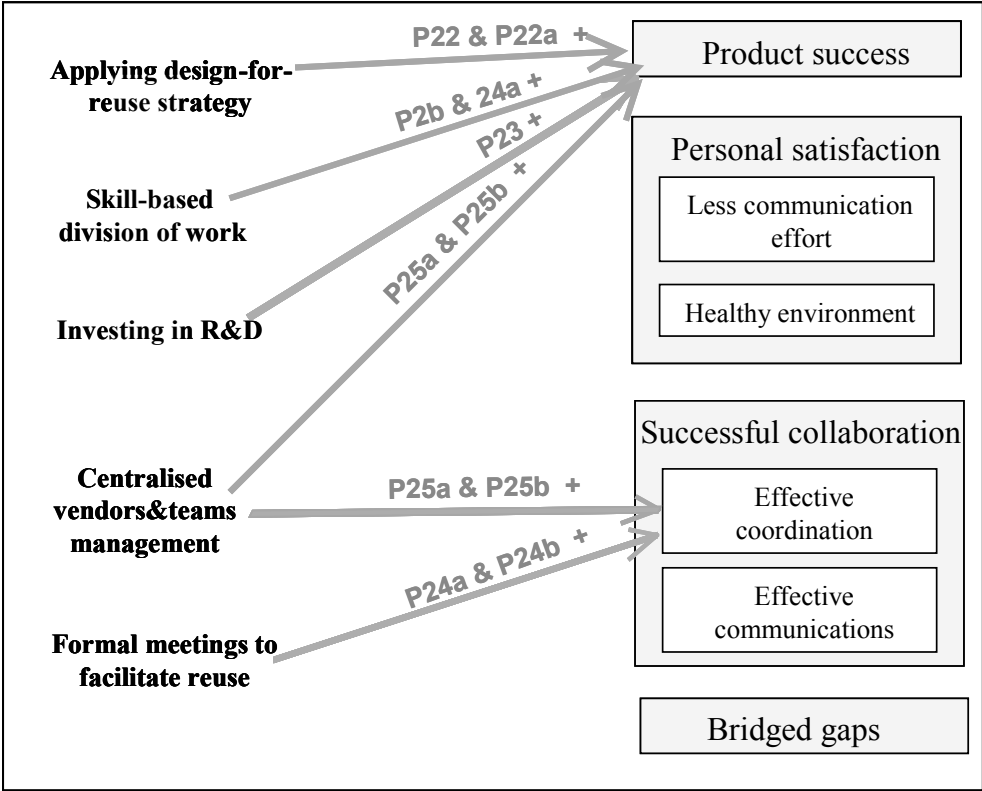
**P25a:** *In GD CBD projects that involve vendors delivering third-party components, centralising coordination of work carried out by all parties involved in the development (internal dispersed development sites and external vendors) will reduce development lifecycle.*

**P25b:** *The more vendors are involved in delivering third-party components, the more important is centralisation of coordination of work carried out by all parties involved in the GD CBD project under one function.*

These propositions are unique to GD CBD projects.

Figure 44 illustrates the relationships between the propositions associated with components management and the categories of success.

**Figure 44: Components management: Propositions**



In this section managerial practices were discussed and compared across the cases. The next section will discuss factors perceived as contributing to success and compare results from the four studied projects.

**9.3 FACTORS CONTRIBUTING TO SUCCESS: CROSS-CASE RESULTS**

The theoretical lens identified four potential factors that may contribute to success in GD CBD: (I) *Inter-site coordination*, (II) *Appropriate tools and technologies*, (III) *Social ties* and (IV) *Knowledge sharing*. The empirical evidence collected in the studied projects supported these four potential factors. Furthermore, one more factor emerged from the data, which is (V) *Components management*.

In order to assess the contribution of the potential factors to success, the analysis of individual cases included mapping of instances in which explicit causal relationships were expressed by an interviewee between managerial practice and success (one of the categories of success). Causal relationships between potential factors and success were derived from the corresponding managerial practices (as explained in Section 4.5). Tables 6, 8 and 10 summarize the contributions of potential factors and managerial practices to success in the LeCroy, SAP and TCS cases respectively.

To compare results across cases, the results from individual cases illustrating the contribution of potential factors to success (rows marked in grey with factors from Tables 6, 8 and 10) were integrated into Table 21<sup>48</sup>.

Taking into account that the Baan case was unsuccessful (according to each category of success), based on the data collected in Baan it is not possible to identify relationships between (lack of) managerial practices and specific success dimensions. Therefore, there are no detailed results from the Baan case that could be included in Table 21.

---

<sup>48</sup> In Table 21 three rows ‘LeCroy’, ‘SAP’ and ‘TCS’ under each factor (I-V) are rows with the same factors from Tables 6, 8 and 10 respectively

**Table 21: Factors contributing to success (per success dimension)**

Factors	Product success	Personal satisfaction		Success. collabor.		Bridged gaps
		Less communication effort	Healthy environment	Effective coordinat.	Effective communic.	
<b>I) Inter-site coordination</b>						
LeCroy		=>	=>	=>	=>	=>
SAP	=>	=>	=>		=>	=>
TCS	=>			=>	=>	=>
<b>II) Appropriate tools and technologies</b>						
LeCroy	=>			=>	=>	=>
SAP		=>	=>		=>	=>
TCS	=>			=>		
<b>III) Social ties</b>						
LeCroy	=>	=>	=>	=>	=>	=>
SAP		=>	=>	=>	=>	=>
TCS	=>	=>	=>	=>		
<b>IV) Knowledge sharing</b>						
LeCroy	=>	=>	=>	=>	=>	=>
SAP		=>	=>	=>	=>	=>
TCS	=>	=>		=>	=>	=>
<b>V) Components management</b>						
LeCroy	=>					
SAP				=>		
TCS	=>			=>		

As can be seen from Table 21, empirical data from all three case studies shows that the four potential factors identified in the theoretical lens indeed contribute to success in GD CBD. Factor (V) *Components management*, which emerged from

the data (mainly in the LeCroy and TCS cases), contributed to success as well, in particular to product success and effective coordination.

#### **9.4 CONCLUSIONS**

In this chapter cross-case analysis and results of the four studied cases were presented and discussed.

First, the similarities and differences between the studied cases were presented. Despite the fact that in many ways the studied projects are similar, there are some differences between the four studied cases, mainly contextual, such as the different countries involved and, consequently, different cultures and different time-zone differences; the different types (and granularity) of components; and different histories (number of years) that remote teams have been working together. These differences between the projects help to explain differences in the results across cases.

Second, managerial practices perceived as important for success in GD CBD in the four studied cases were compared, and propositions that suggest relationships between specific managerial practices and categories of success were formulated. Many of the propositions, in particular propositions regarding inter-site coordination, social ties and knowledge sharing, are not unique to CB but are also relevant in the context of traditional GDSD projects. In regard to CB projects these propositions suggest that, despite the expectations that adoption of CBD in globally distributed projects will allow remote teams to work independently, GD CBD teams that work closely will be more successful (i.e will achieve better project outcomes: shorter time-to-market, lower costs, higher reuse rate), than teams that work independently and do not communicate on a regular basis.

Furthermore, examples of LeCroy and TCS show that in order to succeed in GD CBD and achieve the benefits of components reuse across products, it is important to invest in R&D and apply a design-for-reuse strategy in the early stages of a



project. In LeCroy and TCS these practices facilitated the development of a flexible product architecture and a large pool of reusable components that were later reused in a large number of products.

Finally, the success factors identified in the studied companies were compared. While in the three successful cases managerial practices supporting the five factors (four potential factors identified in the theoretical lens, and one factor that emerged from the data) were evident, in the unsuccessful Baan case only appropriate tools and technologies were in place. The lack of practices supporting non-technical aspects, such as social ties, knowledge sharing and inter-site coordination, further underlines the significance of these factors for success.

This chapter presented cross-case analysis and results. In the next chapter the conclusions of this thesis will be presented.

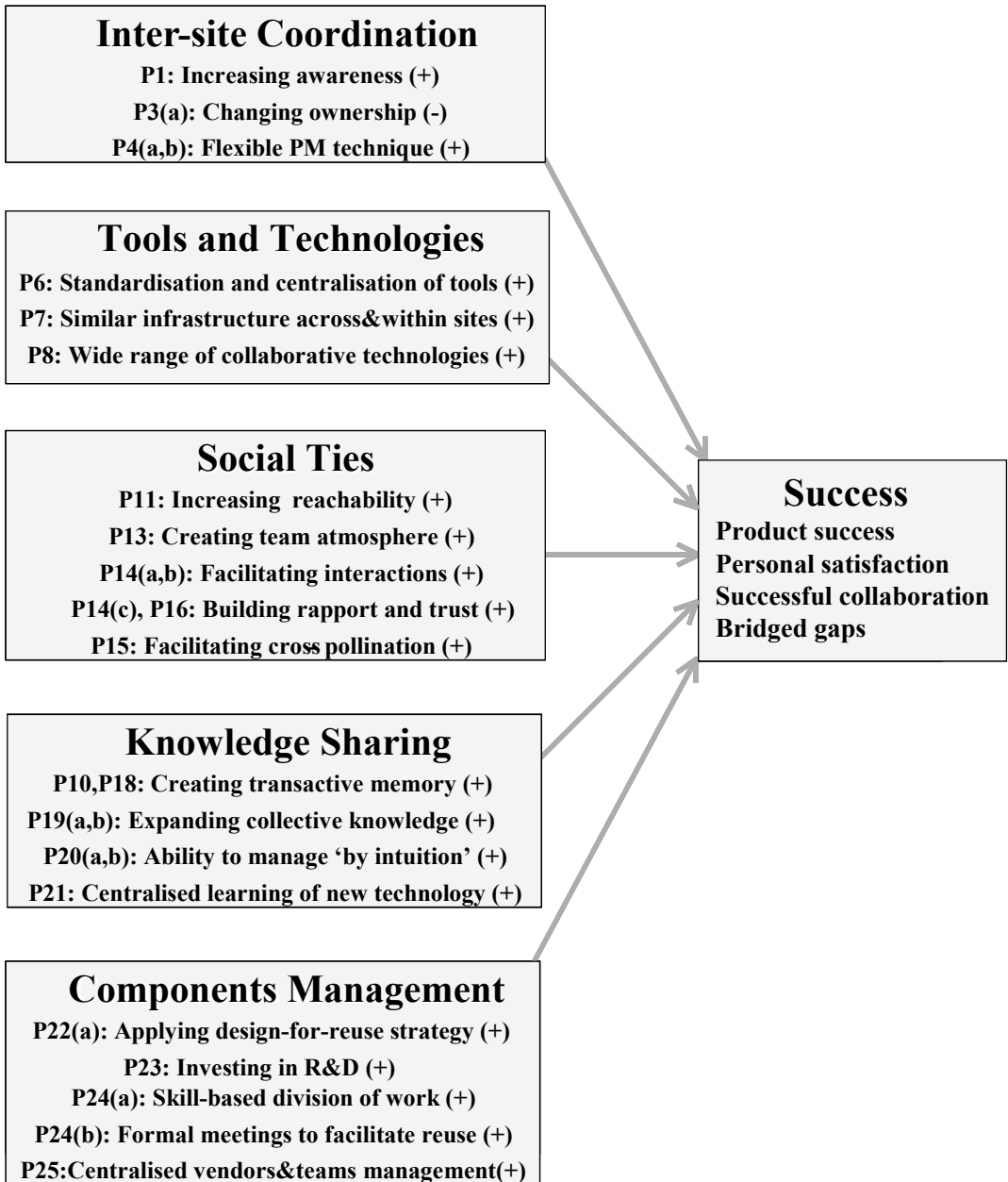
## **CHAPTER 10 CONCLUSIONS**

This chapter summarises the main findings and results of this thesis, and discusses the contributions of this study to the body of knowledge and the management practice. First, the theoretical lens that guided the empirical research is revisited based on the results of the empirical investigation and a theoretical framework is proposed. (Section 10.1). In Section 10.2 the importance of tools and technologies is discussed and compared with the importance of the other four success factors identified in the proposed theoretical framework. In the following Section 10.3, successful managerial practices that illustrate how companies organise and manage CBD in a globally distributed environment are offered. The role of context (e.g. cultures involved, history of working together) in selecting successful managerial practices for GD CBD projects is discussed in Section 10.4. Finally, the theoretical and practical contributions of this research are discussed (Sections 10.5 and 10.6 respectively). This chapter will conclude with the limitations of this study (Section 10.7) and suggestions for future research (Section 10.8).

### **10.1 THEORETICAL LENS: REVISITED**

The potential factors contributing to success, suggested in the initial theoretical lens (Figure 15), can be revisited based on the results of the four case studies discussed above. As a result, a theoretical framework is proposed: it brings together (i) factors contributing to success in GD CBD and (ii) propositions suggesting relationships between specific managerial practices associated with each factor and success, as is presented in Figure 45. (Only propositions that connect managerial practices and success in GD CBD are shown in Figure 45. Propositions related to the contextual characteristics of a project are discussed in Section 10.4).

Figure 45: Theoretical Framework



The theoretical framework in Figure 45 suggests that five factors: (I) *Inter-site coordination*, (II) *Appropriate tools and technologies*, (III) *Social ties*, (IV) *Knowledge sharing* and (V) *Components management* contribute to success in GD CBD teams. The first four factors (I-IV) are supported by IS and OB literature. IS researchers focus on technical and coordination aspects in GDS teams: they suggest that *inter-site coordination* and *tools and technologies* contribute to success in traditional (non-CB) GDS. By contrast, OB researchers focus on the social aspects of collaboration in GD teams: they suggest that *social ties* and *knowledge sharing* contribute to success. The fifth factor, *components management*, emerged from the data.

In regard to GD CBD projects the existing literature is yet very limited (see Section 2.4.6). This literature suggests that (I) *Inter-site coordination* and (II) *Tools and technologies* are considered as contributing to success in GD CBD, while (III) *Social ties* and (IV) *Knowledge sharing* are not discussed in the context of GD CBD teams.

As illustrated in Figure 45, propositions defined as a result of comparison between the four cases (in Chapter 9) are incorporated in the proposed theoretical framework. The propositions suggest relationships between specific managerial practices and categories of success.

## **10.2 THE ROLE OF TECHNOLOGY: TECHNOLOGY ALONE IS NOT ENOUGH**

Technology is crucial for globally distributed teams: without ICT, people at dispersed locations would not be able to connect and collaborate. In the IS literature technology is seen as an enabler and a chief factor that may lead to successful GDS projects (Carmel 1999; Karolak 1999; Herbsleb et al. 2002; van Fenema 2002). However, ***is having the right technology in place enough for a globally distributed team to succeed in GD CBD?***

The importance of tools and technologies can be assessed and compared with the importance of the other four success factors identified in the theoretical framework by comparing the results of the unsuccessful Baan case, where technology was in place but other factors were lacking, with the results from the three successful cases. Table 22 summarises factors that were in place and those that were lacking in all four cases, grouped in successful cases vs. unsuccessful case ('+' – in place, '-' – lacking, 'N/A' – not applicable). The number of managerial practices associated with each factor is shown in brackets (based on the comparison of managerial practices across cases presented in Table 17).

**Table 22: Factors contributing to success**

Project outcome (per case)		Factors that contribute to success				
		Inter-site coordination	Appropriate tools and technologies	Social ties	Knowledge sharing	Components management
<b>Successful:</b>	LeCroy	+ (6)	+ (3)	+ (5)	+ (3)	+ (2)
	SAP	+ (5)	+ (3)	+ (4)	+ (3)	+ (1)
	TCS	+ (6)	+ (3)	+ (4)	+ (3)	+ (4)
<b>Unsuccessful:</b>	Baan	-	+ (3)	-	-	N/A

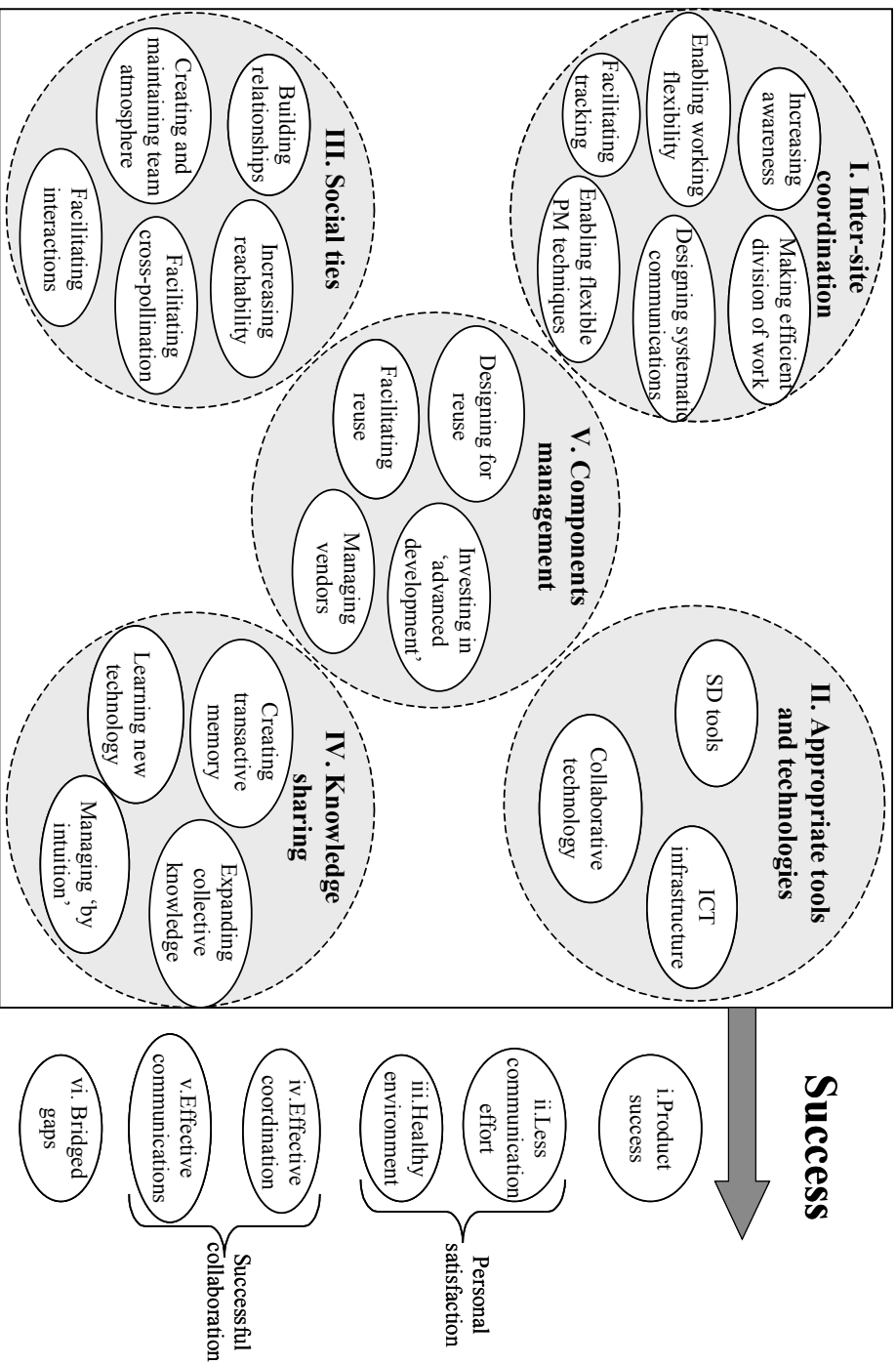
As illustrated in Table 22, while in the three successful cases all five factors identified in the theoretical framework (Figure 45) were evident, in the unsuccessful Baan case only appropriate tools and technologies were in place, and three factors - inter-site coordination, social ties and knowledge sharing, were lacking (the fifth factor – components management – is not relevant as the studied

E-Enterprise project was not component-based). This illustrates that technology alone is not enough to succeed in a globally distributed environment.

### **10.3 HOW COMPANIES ORGANISE AND MANAGE CBD IN A GLOBALLY DISTRIBUTED ENVIRONMENT: SUCCESSFUL MANAGERIAL PRACTICES**

In total 22 managerial practices perceived as important in GD CBD were identified in the studied cases. These practices were classified into groups that focus on different aspects of the management of GD CBD, in accordance with the five factors included in the proposed theoretical framework, and are presented in the form of the concept map in Figure 46. The concept map contains practices identified in all the case studies (as listed in Table 17). The majority of these managerial practices were identified in all successful cases; however, some practices are unique to specific cases, and most of these practices were lacking in the unsuccessful Baan case (as discussed in Section 9.2).

Figure 46: How companies organise and manage GD CBD to be successful



The managerial practices presented in Figure 46 answer the main research question, which is: *how do companies organise and manage CBD in a globally distributed environment to be successful?* Detailed description of these practices is included in the Glossary of Managerial Practices for GD CBD (Appendix 4).

#### **10.4 THE ROLE OF CONTEXT IN SELECTING MANAGERIAL PRACTICES: THE CONTEXT DOES MATTER**

In this research successful managerial practices that show how companies organise and manage GD CBD are identified. These practices are perceived as successful by the interviewees in the studied companies, where application of these managerial practices indeed resulted in successful project outcomes. Nevertheless, would the same managerial practices contribute to success if applied in different companies involved in GD CBD? Not all managerial practices will suit the needs of any GD CBD project. For example, as illustrated in the cross-case analysis (Chapter 9), in all three successful cases practices of efficient division of work were in place. However, specific practices were different between the three companies: in LeCroy work was divided on a skills basis, in SAP by product feature, and in TCS in order to maximise work offshore. These observations lead to the question: *how does the context of GD CBD project organization matter in selecting managerial practices that will be successful for a specific project?*

Several propositions suggested in the previous chapter illustrate how the context matters when selecting successful managerial practices for different GD CBD projects. In particular, the following contextual characteristics were identified:

- *History of working together* defines what strategy for the division of work will be more successful (proposition P2).
- *Existence of relationships such as rapport and trust* have an impact on the choice of a successful strategy for division of work (proposition P2).



- *National culture* has an impact on the following managerial practices:
  - The need to improve content and style of communications, if members of dispersed teams have different cultural backgrounds (proposition P5).
  - The ability to reach the right people varies between cultures, based on the characteristics of a specific culture (proposition P12): for example, in the Indian culture it is usual to contact somebody outside working hours, which is not common in German, Dutch and Swiss cultures.
  - The effort of managers required to build rapport and trust will be different for different cultures (proposition P17): less effort will be required for socially-open cultures, such as India (e.g. in collectivist cultures, according to the Hofstede cultural dimensions).

Therefore, contextual characteristics need to be taken into account when managers select managerial practices to adopt in GD CBD: practices that are successful for one company will not necessarily be successful in another organization, if specific contextual characteristics of these organization are different.

## **10.5 THEORETICAL CONTRIBUTIONS**

This research has studied in depth the phenomenon of GD CBD, which is becoming a promising area, as increasing numbers of companies are setting up software development in a globally distributed environment and at the same time adopt a CBD methodology. Thus, being an emerging area, the management practice of GD CBD is evolving primarily on an *ad hoc* basis.

So far, researchers in the IS field have studied only one aspect of the phenomenon: some have focused on the impact of globalization on the management of traditional (non CB) software development projects, others have focused on the management of CBD in co-located projects. Research into the management of GD

CBD projects that combine these two streams is just emerging and is yet in the early stages. Research on GD CBD has reported that extensive coordination between people working from dispersed locations is required to succeed in GD CBD (Carmel 1999; Alexandersen et al. 2003; Turnlund 2004); other aspects of management were not discussed in this literature. At present, little is known about how to organise and manage GD CBD to be successful. Therefore, the research presented in this thesis advances our knowledge of the management of GD CBD projects, and suggests a more structured (theory-based) approach to the management of such projects. In particular, this thesis makes three main theoretical contributions.

*First*, a theoretical framework that identifies factors contributing to success in GD CBD is proposed (Figure 45). It suggests that (I) *Inter-site coordination*, (II) *Appropriate tools and technologies*, (III) *Social ties*, (IV) *Knowledge sharing* and (V) *Components management* contribute to success in GD CBD.

In particular, interviewees stressed the importance of social ties and knowledge sharing for success. The importance of these two factors has not been identified previously in the IS literature. Therefore, identifying the importance of social ties, such as rapport and trust and knowledge sharing, for success in GD CBD is particularly valuable as it gives a new perspective on the phenomenon of GD CBD, the *importance of social and human aspects in managing GD CBD projects*, which needs to be studied further.

*Second*, 22 managerial practices that illustrate how companies organise and manage CBD in GD environment are offered (Figure 46). In terms of a theoretical contribution, these practices suggest a more structured (theory-based) approach to management of GD CBD projects. These practices support the five success factors included in the proposed theoretical framework (Figure 45).

*Third*, within the IS field, this thesis provides an integrated view which combines three areas of research: (i) IS research on the management of globally distributed development of *traditional* (non-CB) software; (ii) IS research on the management of *co-located* CBD, and (iii) OB research on collaboration in GD teams that examines the importance of social aspects in global collaborations. This thesis connects findings from these three research areas into one integrated framework to study the phenomenon of GD CBD.

## **10.6 PRACTICAL CONTRIBUTIONS: IMPLICATIONS AND LESSONS FOR MANAGERS**

The research presented in this thesis is of high relevance to management practice: it has the following practical contributions and lessons for managers.

*First*, 22 successful managerial practices identified in the studied GD CBD projects are of high relevance to managers. Other companies involved in GD CBD can learn from these practices how to organise and manage GD CBD in their organizations: they can use the concept map with 22 managerial practices (Figure 46) together with the Glossary of Managerial Practices for GD CBD (Appendix 4) as guidelines.

*Second*, specific activities that help to implement the above-mentioned successful managerial practices in actual GD CBD projects are proposed.

As explained in Chapter 4 (Section 4.5), data was analysed on two levels: (i) on a conceptual level, to identify managerial practices (presented in Figure 46); and (ii) on a detailed level, to identify specific activities that helped to implement the managerial practices. Activities identified during detailed analysis are included in the *Checklist for Managers* involved in GD CBD (Table 23): they are grouped into two categories (1) *Before face-to-face (f2f) meeting*, and (2) *After f2f meeting*.

I distinguish between these two stages (before f2f meeting and after f2f meeting) because interviewees from all four companies indicated that their perception of and attitude towards remote colleagues changed dramatically after they met in person, even if only for a short while (as captured in the proposition P14b). Therefore, managers should focus on different sets of activities before team members meet in person and after they meet.

Furthermore, a *Guide to Tools and Technologies for GD CBD* (Table 24) is offered to managers, to help to choose technologies that would match the needs of their organisations. The Guide summarises the main requirements for software development tools and ICT infrastructure, based on the results of all the studied cases (according to the cross-case analysis of tools and technologies, summarised in Tables 18 and 19). Table 2 (adopted from Huis et al. 2002), which describes different types of collaborative technologies, can be included in the Guide.

Moreover, a *Communication Protocol Template* is provided for managers and developers, to help to agree on the rules of communications. The protocol offers recommendations regarding use of collaborative technologies in different situations (these recommendations are based on the cross-case analysis of the use of collaborative technologies summarised in Table 20).

**Table 23: Checklist for managers**

<b>CHECKLIST FOR MANAGERS OF GD CBD</b>
<b>1. Before f2f meeting</b>
<b>1.1 Planning for introductions</b> <i>(tick activities for action)</i> <ul style="list-style-type: none"><li><input type="checkbox"/> virtual f2f meeting</li><li><input type="checkbox"/> introduction of new members</li><li><input type="checkbox"/> kick-off meeting</li><li><input type="checkbox"/> short visit to remote site</li><li><input type="checkbox"/> temporary co-location (long-term stay)</li><li><input type="checkbox"/> social activity</li><li><input type="checkbox"/> team-building exercise</li><li><input type="checkbox"/> show people at remote sites that they are as valuable as the main site</li></ul>
<b>1.2 Design communication processes</b> <i>(tick activities for action)</i> <ul style="list-style-type: none"><li><input type="checkbox"/> set up mini-teams for different functional / technical areas</li><li><input type="checkbox"/> try to reduce the communication paths</li><li><input type="checkbox"/> subsidise language courses (e.g. English)</li><li><input type="checkbox"/> agree on communication rules</li><li><input type="checkbox"/> appoint a contact person for remote teams</li><li><input type="checkbox"/> distribute internal newsletter (e.g. every month)</li><li><input type="checkbox"/> create template for proposals initiating new ideas (e.g. for new product / improvement)</li></ul>

## 2. After f2f meeting

### 2.1 Organise systematic communications *(tick activities for action; decide on frequency of communications)*

- regular (all) managers' meetings, every \_\_\_\_\_
- regular (one-to-one) managers' meetings, every \_\_\_\_\_
- regular meetings with all teams/team members, every \_\_\_\_\_
- regular visits of managers to remote location, every \_\_\_\_\_
- regular reflection sessions, every \_\_\_\_\_

### 2.2 Ensure targeted communications *(tick activities to advise team members)*

- communicate one-to-one (i.e. direct communication, no hierarchy in communications)
- distribute information (without being contacted)
- use synchronous communications

**Table 24: A Guide to Tools and Technologies for managers of GD CBD**

<b>A GUIDE TO TOOLS AND TECHNOLOGIES FOR MANAGERS OF GD CBD</b>
<b>1. Capabilities of Software Development tools</b> <i>(tick required capabilities)</i>
<input type="checkbox"/> Automated management of interdependencies between components and related files
<input type="checkbox"/> Automated building of components that have changed (e.g. every 6-12 hours)
<input type="checkbox"/> Automated testing of components
<input type="checkbox"/> Standardized tools and methods across locations
Centralised tools <i>(you may select more than one from the range of options):</i>
<input type="checkbox"/> Web access
<input type="checkbox"/> Replicated databases
<input type="checkbox"/> Single development environment
<input type="checkbox"/> Central repository/database
<input type="checkbox"/> A Guide that explains how to use tools and methods (make it available for everybody, e.g. on an Intranet)

## 2. ICT Infrastructure *(tick required capabilities)*

Quick access to the network

Shared resources:

Shared databases

Shared server

Shared project repository

Web access

Constant replication of databases over the Web

Centralised access to tools over the Web

Quick and easy connectivity across locations:

Wide range of collaborative technologies available (see different types of collaborative technologies in Table 2)

Phone connection available and easy to use (e.g. internal phone numbers across the globe)

Identical ICT facilities (i.e. network speed, server, applications) for dispersed and co-located teams

## 3. Collaborative technologies

The following *Communication Protocol Template* is intended for all members of a globally distributed teams (not only managers but developers as well). For different types of collaborative tools, the Protocol lists situations/scenarios in which the tool is suitable.



**Communication protocol template: recommended use** *(discuss suggested use of collaborative technologies with your remote counterpart(s) to agree on the rules of communications)*

**Online chat** (suitable for teams with established rapport)

- Short and/or urgent questions

**Phone and teleconferencing**

- Urgent matters
- Updates between managers
- Resolve misunderstandings and conflicts
- Help in fixing bugs

**Application Sharing**

- Help in fixing bugs (e.g. show conditions of failure)
- Knowledge sharing (e.g. show slides during presentation)

**Videoconference**

- Progress meetings between managers
- Major design reviews

**Email**

- Low priority tasks
- Sending source code for small changes
- Clarifications

**Intranet**

- Post internal documents (e.g. specifications, plans, designs, issues)

## 10.7 LIMITATIONS

The conclusions offered in this research are based on an in-depth study of four companies, by applying a qualitative interpretive approach that is often considered as subjective and having limited generalizability (Klein and Myers 1999). Successful managerial practices and factors that contribute to success identified in this research are based to a great extent on the perceptions of interviewees, which may be subjective. To compensate for this subjective source of data, evidence was also collected from internal and external documentation and observations, as suggested by Yin (1994) and Eisenhardt (1989) (Section 4.4), which are considered to be more objective sources, in particular external reports. Applying additional methodological approaches used in positivist research, such as propositions testing and a quantitative survey, may contribute to a further understanding of the phenomenon of GD CBD.

Furthermore, taking into account the fact that national culture was identified as one of the contextual characteristics that need to be considered when selecting managerial practices that would be successful in a specific organisation (Section 10.4), conducting similar case studies that involve different national cultures may reveal new results, unique to the specific cultures. Therefore, conducting more case studies across CBD projects globally distributed across different countries will enable researchers to test the proposed theoretical framework in different cultural settings and will extend the proposed set of managerial practices to include more culture-specific practices.

Finally, it is important to note that many of the successful managerial practices and activities offered to practitioners are expensive (e.g. visits to remote locations, investing in R&D). However, the results of this thesis indicate that investment in these practices in the early stages of a project pays off and is considered beneficial at later stages of the project (as can be seen from the three successful cases

described in this thesis). By contrast, lack of these ‘expensive’ practices in Baan led to project closure.

## **10.8 SUGGESTIONS FOR FUTURE RESEARCH**

The results of this research provide an insight into the technical and social aspects of GD CBD projects that can be further studied in future research. A number of topics can be suggested for a future research agenda.

First, future studies can conduct a survey across the IS industry to test the propositions developed in this research.

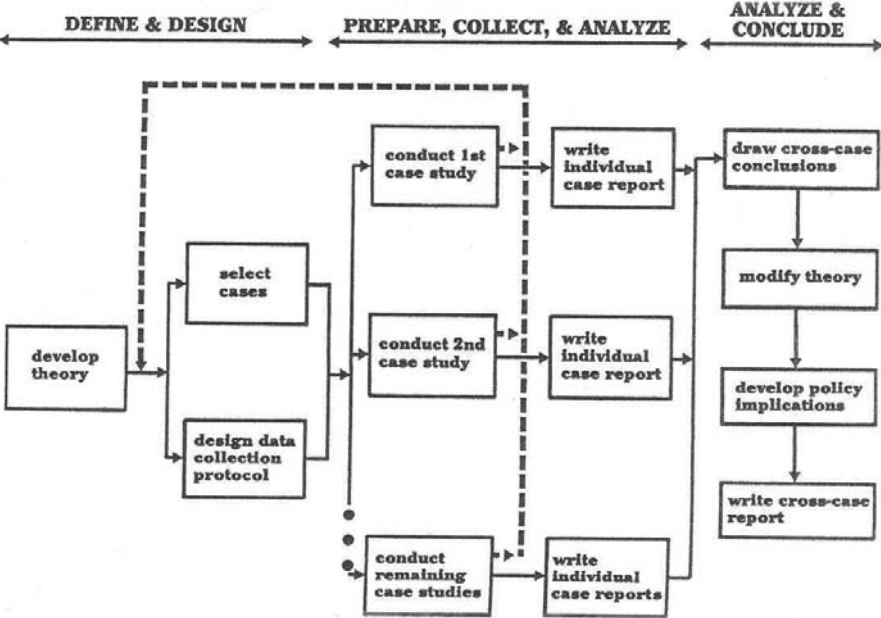
Second, there is a need to investigate the relative importance of each of the five factors included in the proposed theoretical framework. For example, the Baan case illustrated that technology only is not enough to succeed in GD CBD (Section 10.2). More case studies that would offer different combinations of factors that are lacking and factors that are in place will give an opportunity to assess the relative importance of each of the five factors.

Third, the role of social and human aspects in GD CBD can be studied further to investigate the causal relationships between social ties, knowledge sharing and success.

Finally, to study further the phenomenon of GD CBD, exploratory research in different cultural settings and different types of GD CBD, such as Open Source Software development, is needed. Within commercial GD CBD projects, projects that involve more than one company (e.g. outsourcing and joint ventures) need to be explored as well.

APPENDICES

Appendix 1: Replication approach for multiple-case design (adopted from Yin, 1994)



## Appendix 2: Oscilloscopes: general information and products of LeCroy Corporation

‘An oscilloscope is a laboratory instrument commonly used to display and analyze the waveform of electronic signals. In effect, the device draws a graph of the instantaneous signal voltage as a function of time’<sup>49</sup>. Oscilloscopes are used extensively for industrial, scientific and medical purposes (e.g. they are much used for design and testing in high-tech industries, and in research labs in universities). Below are digital oscilloscopes produced by LeCroy Corporation: the WaveMaster Series (left) and the WaveRunner Series (right).



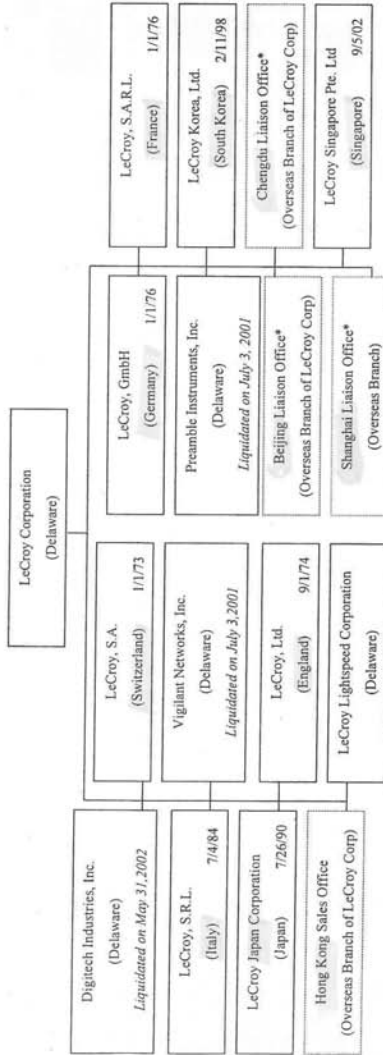
---

<sup>49</sup> Explanation from [www.whatis.com](http://www.whatis.com)

## Appendix 3: LeCroy Corporation: organizational structure

# LeCroy Corporation

For the Fiscal Year Ending June 30, 2003



\* Set up through the Chinese Government's Foreign Enterprise Service Corporation (FESCO). These offices can provide sales, marketing, technical, etc. support to Chinese customers but cannot directly conduct business in China.

### Notes:

Parenthesis indicate Jurisdiction of Incorporation.

## **Appendix 4: Glossary of Managerial Practices for GD CBD**

*(managerial practices organised in alphabetical order)*

### **Building relationships**

Building relationships involves building rapport and trust between remote team members.

*Rapport* is defined as ‘the quality of the relation or connection between interactants, marked by harmony, conformity, accord, and affinity’

(Bernieri et al. 1994)

*Trust* is defined as ‘the willingness of the one person or group to relate to another in the belief that the other’s action will be beneficial rather than detrimental, even though this cannot be guaranteed’ Child (2001).

This practice is related to *social ties*.

### **Collaborative technology**

Collaborative technology covers communication media and collaborative tools. Most commonly used collaborative technologies include email, online chat, phone and teleconferencing, application and desktop sharing, videoconferencing facilities and Intranet (see detailed overview of different types of collaborative technologies in Table 2 adopted from Huis et al. 2002).

This practice is related to *appropriate tools and technologies*.

### **Creating and maintaining team atmosphere**

Creating and maintaining a team atmosphere implies making sure that all remote teams/team members are ‘plugged’ into the project/company: in particular, it is relevant for members of offshore teams.

This practice is related to *social ties*.

### **Creating transactive memory among dispersed team members**

Transactive memory is defined as the set of knowledge possessed by group members coupled with an awareness of who knows what (Wegner 1987). Transactive memory of a globally distributed team implies that team members know the composition of a remote team (who people are, their roles) and know the areas of expertise of their remote counterparts. This practice is related to *knowledge sharing*.

### **Designing for reuse**

This practice aims to maximise reuse of software components across a number of products in the long term. This involves analysis and long term planning for future products and product families (i.e. identifying common functionalities), and making strategic decisions about the granularity level of components.

This practice emphasizes the reuse during the *planning* stage of a project, before the actual development has started. It is related to *components management*.

### **Designing systematic communications**

This practice includes organising frequent communications and designing rules aiming to make communications more effective. It includes:

- (i) Scheduling systematic and frequent communications, such as regular teleconferences between software managers in dispersed locations; videoconferences with all team members (e.g. every one or two months).
- (ii) Communicating directly to reach an appropriate person. i.e. no



hierarchy in communications.

(iii) Improving style and content of communications to achieve better understanding (and prevent conflict and misunderstanding) between remote counterparts.

This practice is related to *inter-site coordination*.

### **Enabling flexible Project Management (PM) techniques**

Flexible PM techniques help to accommodate everyday dynamics.

They include:

- On a macro level: planning of major project phases and deliverables, and setting up clear objectives for a dispersed team
- On a micro level: flexible and not too detailed planning (2-3 week milestones).

This practice is related to *inter-site coordination*.

### **Enabling working flexibility**

Supporting working flexibility implies providing (i) flexible working conditions e.g. working from home, and (ii) flexible working hours, in order to accommodate personal circumstances of team members, to make their working environment more convenient and comfortable.

This practice is related to *inter-site coordination*.

### **Expanding collective knowledge of the dispersed team**

Collective knowledge comprises elements of knowledge that are common to all members of an organisation (Grant 1996).

In the case of GD CBD projects, the 'organisation' involves all people participating in the globally distributed project from their remote locations. Therefore, the collective knowledge of a dispersed team includes knowledge of the national culture of remote counterparts,

collective knowledge of the overall product (beyond a specific area an individual team member is working on) and common technical knowledge.

This practice is related to *knowledge sharing*.

### **Facilitating cross-pollination**

Cross-pollination implies that people from the one group spend significant amounts of time in the other group (other location) and vice versa.

This practice is related to *social ties*.

### **Facilitating interactions**

Facilitating interactions between people at remote locations includes:

- (i) facilitating personal face-to-face interactions (in particular between key people for decision-making) and
- (ii) organising regular and frequent interactions over distance.

This practice is related to *social ties*.

### **Facilitating reuse**

This practice implies facilitating the reuse of knowledge and reuse of components across dispersed locations. It includes identifying an opportunity to reuse knowledge and/or software components (applications) developed by dispersed teams. Reuse can be facilitated on two levels: within one product and across different products of the same product family.

This practice emphasizes reuse in *ongoing* projects. It is related to *components management*.

### **Facilitating tracking of bugs and development tasks**

Tracking possibility means (i) having constantly updated status about the stages in fixing a bug, or progress in a task, and (ii) knowing who is responsible for fixing the bug, or completing the task.

This practice is related to *inter-site coordination*.

### **ICT infrastructure**

An ICT infrastructure enables connection between all remote sites. It includes Internet, WAN, server and applications pool, how resource shares are set up (i.e. sharing of databases, server, project repository), conferencing tools, and network speed and bandwidth. Furthermore, it includes capabilities aiming to support security requirements, such as firewalls and access rights.

This practice is related to *appropriate tools and technologies*.

### **Increasing awareness**

This practice involves increasing awareness of (i) what is going on in the company and the project, (ii) progress made by remote teams, (iii) remote team members (the team composition, culture of remote counterparts and cultural differences), and (iv) the environment at a remote site, e.g. ICT and tools available for remote teams.

This practice is related to *inter-site coordination*.

### **Increasing reachability**

Increasing reachability implies making it easier to reach the right people at a remote location, in particular, (i) knowing whom to contact (i.e. who is the person who has knowledge of a certain domain or issue), and (ii) knowing who is available on the given day and time.

This practice is related to *social ties*.

### **Investing in ‘advanced development’**

Investing in ‘advanced development’ implies that a development of a new CB product is treated not as a typical product development project, in which product requirements are defined in the very beginning, but as a research (i.e. R&D) project.

‘Advanced development’ includes learning about available technologies, and conducting a feasibility study aiming to test whether or not a ‘proof of concept’ for the product can be achieved by applying available technology(ies).

This practice is related to *components management*.

### **Learning new technology**

Learning a new technology includes (i) learning a specific component technology used for developing a CB product, and (ii) learning of the CB product, the principles and logic that it is based upon.

This practice is related to *knowledge sharing*.

### **Making efficient division of work**

Efficient division of work involves strategies that software managers follow (i) to divide work between globally distributed teams (e.g. by skills or by product features / components), as well as (ii) to divide specific assignments (tasks) and responsibilities between individual team members at remote locations (e.g. division of technical and ‘social’ responsibilities that include establishing reporting channels across the globe). This practice also includes the approach regarding ownership of work packages – whether ownership stays with the same teams or is shifted between dispersed teams during a project lifecycle.

This practice is related to *inter-site coordination*.

### **Managing ‘by intuition’**

Management ‘by intuition’ is based on catching signals and sensing that something is working or not working properly. It implies having an intuitive awareness of the situation (environment) and of remote counterparts (in particular, managers of remote teams and key members).

This practice is related to *knowledge sharing*.

### **Managing vendors**

This practice implies managing vendors providing third-party components: this includes selecting vendors, agreeing on specifications of the components (e.g. functionality and interfaces), and deadlines for components’ delivery.

This practice is related to *components management*.

### **Software Development tools**

Software Development tools include tools for development and management of components, configuration and version management tools, tools for testing and tracking bugs. This practice also includes software development methods and processes.

This practice is related to *appropriate tools and technologies*.

## Appendix 5: Baan product version compatibility matrix (internal document)

### Baan E-Enterprise Product Matrix E-Common

Web Server Configuration	Browsers	IE 4.01 SP1	IE 4.01 SP2	IE 5.x	IE 6.x	NS 4.72	NS 6.x	NS 6.0	NS 6.1	NS 6.2	IE 5.0	IE 5.5	IE 6.0
E-Sales 2.0		Yes 1)	No 2)	No 2)									
E-Collaboration 2.0		Yes 1)	No 2)	No 2)									
E-Procurement 2.0		Yes 2)	Yes	Yes									
E-Sales 2.1		Yes 2)	Yes	Yes									
E-Collaboration 2.1		Yes 2)	Yes	Yes									
E-Procurement 2.1		Yes 2)	Yes	Yes									
E-Service 2.0		Yes 2)	Yes	Yes									
E-Service Remote 2.0		No	No	Yes									
E-Sales 2.2		Yes 2)	Yes	Yes									
E-Collaboration 2.2		Yes 2)	Yes	Yes									

1) This product contains E-Common 2.0 which works only with MDAC 2.0 SP2 (which is part of IE 4.01 SP1)

2) This product contains E-Common 2.1(\*) which works only with MDAC 2.1 (which is part of IE 4.01 SP2)

When used with IE 4.01 SP1 MDAC 2.1 needs to be installed manually or MS SQL 7 has to be installed on the webserver (contains MDAC 2.1).

### Web Server Configuration Databases, MS Technology

	MS SQL 7.0	Office2000	ADSI 2.5	Frontpage2	Site Serv.SP2	Site Serv.SP1
E-Sales 2.0	No 3)	No	Yes 4)	No	Yes 5)	
E-Collaboration 2.0	No 3)	No	Yes 4)	No	Yes 5)	
E-Procurement 2.0	Yes	Yes	Yes 4)	No	Yes 5)	
E-Sales 2.1	Yes	Yes	Yes 4)	No	Yes 5)	
E-Collaboration 2.1	Yes	Yes	Yes 4)	No	Yes 5)	
E-Procurement 2.1	Yes	Yes	Yes 4)	No	Yes 5)	
E-Service 2.0	Yes	Yes	Yes 4)	No	Yes 5)	
E-Service Remote 2.0	Yes	Yes	Yes 4)	No	Not test	
E-Sales 2.2	Yes	Yes	Yes 4)	No	Yes 5)	
E-Collaboration 2.2	Yes	Yes	Yes 4)	No	Yes 5)	

3) Because MS SQL 7 installs MDAC 2.1

4) Only allowed in combination with ADSI 2.5

### Web Server Configuration Proprietary technology

	OW2.1	OW2.2	OW2.2.1	OW3.0	BCLM 1.0
E-Sales 2.0					
E-Collaboration 2.0					
E-Procurement 2.0					
E-Sales 2.1					
E-Collaboration 2.1					
E-Procurement 2.1	yes				
E-Service 2.0					
E-Service Remote 2.0					
E-Sales 2.2					
E-Collaboration 2.2					

### Applications and database on the same server

E-Sales 2.0	no
E-Collaboration 2.0	no
E-Procurement 2.0	yes
E-Sales 2.1	yes
E-Collaboration 2.1	yes
E-Procurement 2.1	yes
E-Service 2.0	yes
E-Service Remote 2.0	yes
E-Sales 2.2 LA/GA	yes
E-Collaboration 2.2	yes

### Applications running with which E-Common version 1)

E-Sales 2.0	E-Common 2.0
E-Collaboration 2.0	E-Common 2.0
E-Procurement 2.0	E-Common 2.1
E-Sales 2.1	E-Common 2.2
E-Collaboration 2.1	E-Common 2.2
E-Procurement 2.1	E-Common 2.3.0
E-Service 2.0	E-Common 2.3.1
E-Service Remote 2.0	E-Common 2.3.1
E-Sales 2.2 LA	E-Common 2.3.1
E-Sales 2.2 GA	E-Common 2.3.1 SP1
E-Sales 2.2 SP1	E-Common 2.3.1 SP2
E-Collaboration 2.2	E-Common 2.3.1 SP2
E-Procurement 2.1 SP1	E-Common 2.3.1 SP3

1) Applications which are running on the same E-Common version are able to run together on one server.

2) Applications which are running on E-Common 2.2 or higher are able to run together on one server

### Applications running with Extensions

	Automotive VR(A&D)	Status
E-Sales 2.1	NASCAR SCH1	tested
E-Collaboration 2.1	NASCAR SCH1	tested
E-Procurement 2.1	NASCAR SCH1	tested
E-Service 2.0		
E-Service 2.0 Remote		
E-Sales 2.2 LA	NASCAR SCH1	expected to work, not tested
E-Collaboration 2.2	NASCAR SCH1	expected to work, not tested

## Appendix 6: Baan connectivity pack (internal document)

### Connectivity Packs

Connectivity Pack 2.0.0	Triton 3.1	Baan IVb2	Baan IVc	Baan Vb	Baan Vc
E-Sales 2.0	no	yes	yes	no	no
E-Collaboration 2.0	no	yes	yes	no	no
E-Procurement 2.0	no	yes	yes	no	no
E-Sales 2.1	no	no	no	no	no
E-Collaboration 2.1	no	no	no	no	no
E-Procurement 2.1	no	no	no	no	no
E-Service 2.0	no	no	no	no	no
E-Service Remote 2.0	no	no	no	no	no

Connectivity Pack 2.1.0	Triton 3.1	Baan IVb2	Baan IVc	Baan Vb	Baan Vc
E-Sales 2.0	no	yes	yes	no	no
E-Collaboration 2.0	no	yes	yes	no	no
E-Procurement 2.0	no	yes	yes	no	no
E-Sales 2.1	no	yes	yes	yes	no
E-Collaboration 2.1	no	yes	yes	yes	no
E-Procurement 2.1	no	no	no	no	no
E-Service 2.0	no	no	no	no	no
E-Service Remote 2.0	no	no	no	no	no

Connectivity Pack 2.1.1	Triton 3.1	Baan IVb2	Baan IVc	Baan Vb	Baan Vc
E-Sales 2.0	no	no	no	no	no
E-Collaboration 2.0	no	no	no	no	no
E-Procurement 2.0	no	no	no	no	no
E-Sales 2.1	no	yes	yes	yes	no
E-Collaboration 2.1	no	yes	yes	yes	no
E-Procurement 2.1	no	yes	yes	yes	no
E-Sales 2.2	no	yes	yes	yes	no
E-Service 2.0	no	no	no	yes	no
E-Service Remote 2.0	no	no	no	yes	no

### iPack E-Sales 2.2 SP1 - Vc

Connectivity Pack 2.1.1	Triton 3.1	Baan IVb2	Baan IVc	Baan Vb	Baan Vc
E-Sales 2.0	no	no	no	no	no
E-Collaboration 2.0	no	no	no	no	no
E-Procurement 2.0	no	no	no	no	no
E-Sales 2.1	no	yes	yes	yes	no
E-Collaboration 2.1	no	yes	yes	yes	no
E-Procurement 2.1	no	yes	yes	yes	no
E-Sales 2.2	no	yes	yes	yes	no
E-Service 2.0	no	no	no	yes	no
E-Service Remote 2.0	no	no	no	yes	no
E-Collaboration 2.2	no	yes	yes	yes	no

## EXECUTIVE SUMMARY

Globally Distributed Component-Based Development (GD CBD) is expected to become a promising area, as increasing numbers of companies are setting up software development in a globally distributed environment and at the same time are adopting CBD methodologies. This process of globalization and adoption of CBD methodology has introduced potential benefits as well as new challenges in the management of software projects.

On the one hand, it is expected that adoption of CBD will further facilitate globally distributed development of software products, as happened in industries such as aeronautics, automotive, electronics and computers hardware, where CB architectures have been successfully used for setting up globally distributed design and production. Within the software industry, it is suggested that components could be developed in parallel independently by teams located in the same building or at remote locations. It has been argued that CBD enables each site to take ownership of particular components and work on them independently without much need for inter-site communication and coordination (Carmel 1999; Colbert et al. 2001; Repenning et al. 2001).

On the other hand, research on co-located CBD projects has reported difficulties associated with the management of CBD projects, such as lack of stable standards, lack of reusable components, and problems related to the granularity and generality of components (Vitharana 2003). In the light of these problems, achieving the true potential of CBD, which is mainly about reusing components, is challenging even in co-located CBD projects (Crnkovic and Larsson 2002). Globally distributed organizations may face the above-mentioned and additional challenges (caused by geographical, time-zone and cultural differences) when adopting the practice of CBD.

Being an emerging area, the management of GD CBD has evolved primarily on an *ad hoc* basis. At present, little is known about how to successfully organise and



manage GD CBD. To fill this gap, this research explores the management of GD CBD and reveals factors that contribute to success in GD CBD projects. Data are drawn from several successful GD CBD projects at LeCroy, SAP and TCS, compared with one unsuccessful project at Baan. The results suggest that inter-site coordination, appropriate tools and technologies, social ties, knowledge sharing and components management are the main factors that contribute to success in GD CBD. In particular, interviewees stressed the importance of social ties and knowledge sharing for success. The importance of these two factors has not been identified previously in the IS literature.

Furthermore, a framework assisting managers to organize and manage CBD in globally distributed environments is offered. It includes 22 managerial practices that describe how companies organise and manage CBD in a globally distributed environment; a checklist for managers that lists specific activities that help to implement the above-mentioned managerial practices; and a guide for tools and technologies. In terms of a theoretical contribution, these practices suggest a more structured (theory-based) approach to the management of GD CBD projects.

Moreover, the results of this research reveal that, despite the expectations that adoption of CBD in globally distributed projects will allow remote teams to work independently, GD CBD teams that work closely will be more successful (i.e will achieve better project outcomes: shorter time-to-market, lower costs, higher reuse rate) than teams that work independently and do not communicate on a regular basis.

Lastly, the examples of LeCroy and TCS show that in order to succeed in GD CBD and achieve the benefits of components reuse across products, it is important to apply design-for-reuse strategy and invest in R&D in the early stages of a project. In LeCroy and TCS these practices facilitated the development of a flexible product architecture and a large pool of reusable components that were later reused in a large number of products.

## REFERENCES

- Adler, P. S. and B. Borys (1996). 'Two types of bureaucracies: enabling and coercive'. *Administrative Science Quarterly* **41**: pp.61-89.
- Alexandersen, C., K. Kumar and J. Van Hillegersberg (2003). 'Bank-in-a-Box: Skandia 's a Agile and Customizable Financial Services Platform'. <http://www.simnet.org/>, Society for Information Management (SIM).
- Andres, H. P. (2002). 'A Comparison of Face-to-face and Virtual Software Development Teams'. *Team Performance Management* **8**(1/2): pp.39-48.
- Arino, A., J. de la Torre and P. S. Ring (2001). 'Relational Quality: Managing Trust in Corporate Alliances'. *California Management Review* **44**(1): pp.109-131.
- Ba, S. (2001). 'Establishing Online Trust Through a Community Responsibility System'. *Decision Support Systems* **31**: pp.323-336.
- Baker, S., M. Spiro and S. Hamm (2000). 'The fall of Baan'. Business Week.
- Bass, L., C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord and K. Wallnau (2000). 'Volume I: Market Assessment of Component-Based Software Engineering', Carnegie Mellon Software Engineering Institute (SEI).
- Battin, R. D., R. Crocker and J. Kreidler (2001). 'Leveraging Resources in Global Software Development'. *IEEE Software*(March/April): pp.70-77.
- Baumard, P. (1999). *Tacit Knowledge in Organizations*. London, SAGE Publications.
- Beath, C. M. and W. J. Orlikowski (1994). 'The Contradictory Structure of Systems Development Methodologies: Deconstructing the IS-User Relationship in Information Engineering'. *Information Systems Research* **5**(4): pp.350-377.
- Bechky, B. A. (2003). 'Sharing Meaning Across Occupational Communities: The Transformation of Understanding on a Production Floor'. *Organization Science*. **14**: pp.312-330.
- Bernieri, F. J., J. M. Davis, R. Rosenthal and K. C.R. (1994). 'Interactional Synchrony and Rapport: Measuring Synchrony in Displays Devoid of Sound and Facial Affect'. *Personality and Social Psychology Bulletin* **20**: pp.303-311.

- Beynon-Davies, P., C. Carne, H. Mackay and D. Tudhope (1999). 'Rapid application development (RAD): An empirical review'. *European Journal of Information Systems* **8**(3): pp.211-223.
- Brooks, F. P. (1987). 'No Silver Bullet'. *Developer productivity*(November): pp.39-48.
- Carmel, E. (1999). *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, NJ, Prentice-Hall P T R.
- Carmel, E. (2003a). 'Taxonomy of New Software Exporting Nations'. *The Electronic Journal on Information Systems in Developing Countries (EJISDC)* **13**(May).
- Carmel, E. (2003b). 'The New Software Exporting Nations: Impacts on National Well Being Resulting from their Software Exporting Industries'. *The Electronic Journal on Information Systems in Developing Countries (EJISDC)* **13**(May).
- Carmel, E. and R. Agarwal (2001). 'Tactical Approaches for Alleviating Distance in Global Software Development'. *IEEE Software*(March/April): pp.22-29.
- Carmel, E. and R. Agarwal (2002). 'The Maturation of Offshore Sourcing of Information Technology Work'. *MIS Quarterly Executive* **1**(2): pp.65-77.
- Carmel, E., R. Whitaker and J. F. George (1993). 'Participatory Design and Joint Application Design: a transatlantic comparison'. *Communications of the ACM* **36**(6): pp.40-48.
- Cheng, L., C. R. B. De Souza, S. Hupfer, J. Patterson and S. Ross (2004). 'Building Collaboration into IDEs'. *Queue* **1**(9): pp.40-50.
- Child, J. (2001). 'Trust - The Fundamental Bond in Global Collaboration'. *Organizational Dynamics* **29**(4): pp.274-288.
- Colbert, R. O., D. S. Compton, R. L. Hackbarth, J. D. Herbsleb, L. A. Hoadley and G. J. Wills (2001). 'Advanced Services: Changing How We Communicate.' *Bell Labs Technical Journal* **6**(1): pp.211-228.
- Crampton, C. D. (2001). 'The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration'. *Organization Science* **12**(3): pp.346-371.
- Crnkovic, I. and M. Larsson (2002). 'Challenges of Component-Based Development'. *The Journal of Systems and Software*(61): pp.201-212.
- Crowston, K. and E. E. Kammerer (1998). 'Coordination and collective mind in software requirements development'. *IBM Systems Journal* **37**(2): pp.227-245.

- Crowston, K. and B. Scozzi (2002). 'Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development'. *IEE Proceedings - Software* **149**(1).
- Damian, D. (2002). 'The Study of Requirements Engineering in Global Software Development: as Challenging as Important'. *Workshop on Global Software Development, part of the International Conference on Software Engineering (ICSE)*, Orlando, Florida, USA.
- Damian, D., J. Chisan, P. Allen and B. Corrie (2003). 'Awareness Meets Requirements Management: Awareness Needs in Global Software Development'. *Workshop on Global Software Development, part of the International Conference on Software Engineering (ICSE)*, Portland, Oregon, USA.
- Datar, S., C. Jordan, S. Kekre and K. Srinivasan (1997). 'New Product Development Structures and Time-to-Market'. *Management Science* **43**(4): pp.452-464.
- DeSanctis, G. and R. B. Gallupe (1987). 'A Foundation for the Study of Group Decision Support Systems'. *Management Science* **33**(5): pp.589-609.
- Donnellon, A., B. Gray and M. G. Bougon (1986). 'Communication, Meaning, and Organized Action'. *Administrative Science Quarterly*. **31**: pp.43-55.
- D'Souza, D. F. and A. C. Wills (1999). *Objects, components, and frameworks with UML: the Catalysis approach*. Amsterdam, Addison- Wesley.
- Dyer, J. H. (2001). 'How to Make Strategic Alliances Work'. *MIT Sloan Management Review* **42**(4): pp.37-43.
- Ebert, C. and P. De Neve (2001). 'Surviving Global Software Development'. *IEEE Software*(March/April): pp.62-69.
- Eisenhardt, K. M. (1989). 'Building Theories from Case Study Research'. *Academy of Management Review* **14**(4): pp.532-550.
- Espinosa, A. and E. Carmel (2003). 'Modeling Coordination Costs Due to Time Separation in Global Software Teams'. *Workshop on Global Software Development, part of the International Conference on Software Engineering (ICSE)*, Portland, Oregon, USA.
- Espinosa, A., J. N. Cummings, J. M. Wilson and B. M. Pearce (2003). 'Team Boundary Issues Across Multiple Global Firms'. *Journal of Management Information Systems* **19**(4): pp.157-190.

- Evaristo, R. and P. C. van Fenema (1999). 'A Typology of Project Management: Emergence and Evolution of New Forms'. *International Journal of Project Management* **17**(5): pp.275-281.
- Faraj, S. and L. Sproull (2000). 'Coordinating Expertise in Software Development Teams'. *Management Science* **46**(12): pp.1554-1568.
- Firesmith, D. and B. Henderson-Sellers (2001). *The OPEN Process Framework*, Addison Wesley.
- Gabarro, J. J. (1990). 'The development of working relationships'. *Intellectual teamwork: Social and technological foundations of cooperative work*. J. Galegher, R. E. Kraut and C. Egidio. Hillsdale, New Jersey, Lawrence Erlbaum Associates: pp.70-110.
- Gallivan, M. J. (2001). 'Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies.' *Information Systems Journal* **11**(4): pp.227-304.
- Graham, I., B. Henderson-Sellers and H. Younessi (1997). *The Open Process Specification (Open Series)*, Addison Wesley.
- Granovetter, M. S. (1973). 'The Strength of Weak Ties'. *American Journal of Sociology* **78**(6): pp.1360-1380.
- Grant, R. M. (1996). 'Toward a knowledge-based theory of the firm'. *Strategic Management Journal* **17**(Winter): pp.109-122.
- Gremler, D. D. and K. P. Gwinner (2000). 'Customer-Employee Rapport in Service Relationships'. *Journal of Service Research* **3**(1): pp.82-104.
- Grinter, R. E. (1995). 'Using a Configuration Management Tool to Coordinate Software Development'. *Conference on Organizational Computing Systems*, Milpitas, CA, USA.
- Grinter, R. E. (1998). 'Recomposition: Putting It All Back Together Again'. *Conference on Computer-Supported Cooperative Work*, Seattle, Washington, USA.
- Grinter, R. E. (1999). 'Systems Architecture: Product Design and Social Engineering'. *International Joint Conference on Work Activities Coordination and Collaboration*, San Francisco, CA, USA, ACM Press.
- Grinter, R. E., J. D. Herbsleb and D. E. Perry (1999). 'The Geography of Coordination: Dealing with Distance in R&D Work'. *International ACM SIGGROUP Conference on Supporting Group Work (Group 99)*, Phoenix, Arizona, USA, ACM Press.

- Handel, M. and J. D. Herbsleb (2002). 'What is Chat Doing in the Workplace?' *Conference on Computer-Supported Cooperative Work*, New Orleans, LA, USA.
- Hansen, M. T. (2002). 'Knowledge Networks: Explaining Effective Knowledge Sharing in Multiunit Companies'. *Organization Science* **13**(3): pp.232-248.
- Herbsleb, J. D., D. L. Atkins, D. G. Boyer, M. Handel and T. A. Finholt (2002). 'Introducing Instant Messaging and Chat into the Workplace'. *Conference on Computer-Human Interaction*, Minneapolis, MN, USA.
- Herbsleb, J. D. and R. E. Grinter (1999). 'Architectures, Coordination, and Distance: Conway's Law and Beyond'. *IEEE Software*: pp.63-70.
- Herbsleb, J. D. and A. Mockus (2003). 'An Empirical Study of Speed and Communication in Globally-Distributed Software Development'. *IEEE Transactions on Software Engineering* **29**(6): pp.1-14.
- Herbsleb, J. D., A. Mockus, T. A. Finholt and R. E. Grinter (2000). 'Distance, Dependencies, and Delay in Global Collaboration'. *Conference on Computer Supported Cooperative Work*, Philadelphia, Pennsylvania, USA.
- Herbsleb, J. D. and D. Moitra (2001). 'Global Software Development'. *IEEE Software*(March-April): pp.16-20.
- Herrera, M. (2002). 'Globally Distributed Software Development: The need for integration of Object Oriented CASE tools with Groupware' (Master Thesis). Department of Decision and Information Sciences. *Erasmus University*: 120.
- Herzum, P. and O. Sims (2000). *Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise*. New York, Wiley Computer Publishing.
- Hirschheim, R. and H. K. Klein (1994). 'Realizing emancipatory principles in information systems development'. *MIS Quarterly* **18**(1): pp.83-109.
- Hoegl, M. and H. G. Gemuenden (2001). 'Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence'. *Organization Science* **12**(4): pp.435-449.
- Hofstede, G. (1993). 'Cultural constraints in management theories'. *Academy of Management Executive* **7**(1): pp.81-94.

- Homann, U., M. Rill and A. Wimmer (2004). 'Flexible Value Structures in Banking'. *Communications of the ACM* **47**(5): pp.34-36.
- Huang, J. C., S. Newell, R. D. Galliers and S.-L. Pan (2003). 'Dangerous Liaisons? Component-Based Development and Organizational Subcultures'. *IEEE Transactions on Engineering Management* **50**(1): pp.89-99.
- Huis, M. A. A., A. J.H.E. and M. Soekijad (2002). 'ICT Facilitation of Distributed Groups and Communities'. *Building Blocks for Effective Telematics Application Development and Evaluation*, Delft University of Technology: pp.39-46.
- Jarvenpaa, S. L., K. Knoll and D. E. Leidner (1998). 'Is anybody out there? Antecedents of trust in global virtual teams'. *Journal of MIS* **14**(4): pp.29-64.
- Jarvenpaa, S. L. and D. E. Leidner (1998). 'Communication and Trust in Global Virtual Teams'. *Journal of Computer-Mediated Communication (Online at <http://www.ascusc.org/jcmc>)* **3**(4).
- Karolak, D. W. (1999). *Global Software Development: Managing Virtual Teams and Environments*, IEEE Computer Society.
- Kim, S. D. (2002). 'Lessons Learned from a Nationwide CBD PROMOTION PROJECT'. *Communications of the ACM* **45**(10): pp.83-87.
- Klein, H. K. and M. D. Myers (1999). 'A set of principles for conducting and evaluating interpretive field studies in Information Systems'. *MIS Quarterly* **23**(1): pp.67-94.
- Kobitzsch, W., D. Rombach and R. L. Feldmann (2001). 'Outsourcing in India'. *IEEE Software*(March/April): pp.78-86.
- Kobylynski, R., O. Creighton, A. H. Dutoit and B. Brugge (2002). 'Building Awareness in Global Software Engineering: Using Issues as Context'. *Workshop on Global Software Development, part of the International Conference on Software Engineering (ICSE)*, Orlando, Florida, USA.
- Kotlarsky, J. and I. Oshri (2005). 'Social ties, knowledge sharing and successful collaboration in globally distributed system development projects'. *European Journal of Information Systems* **14**(1): pp.37-48.
- Kraut, R. E. and L. A. Streeler (1995). 'Coordination in Software Development'. *Communications of the ACM* **38**(3): pp.69-81.
- Kumar, K. and L. P. Willcocks (1996). 'Offshore Outsourcing: A Country Too Far?' *European Conference on Information Systems*, Lissabon, Portugal.

- Kunda, D. and L. Brooks (2000). 'Assessing organisational obstacles to component-based development: a case study approach'. *Information and Software Technology* **42**: pp.715-725.
- Lee, A. S. (1991). 'Integrating Positivist and Interpretive Approaches to Organizational Research'. *Organization Science* **2**(4): pp.342-365.
- Lyengar, P. (2004). 'Application Development Is More Global Than Ever', Gartner: pp.2.
- Majchrzak, A., R. E. Rice, N. King, A. Malhotra and S. Ba (2000). 'Computer-mediated inter-organizational knowledge-sharing: Insights from a virtual team innovating using a collaborative tool'. *Information Resources Management Journal* **13**(1): pp.44-54.
- Malhotra, A., A. Majchrzak, R. Carman and V. Lott (2001). 'Radical Innovation Without Collocation: A Case Study at Boeing-Rocketdyne'. *MIS Quarterly* **25**(2): pp.229-249.
- Massey, A. P., M. M. Montoya-Weiss and T. M. O'Driscoll (2002). 'Knowledge Management in Pursuit of Performance: Insights from Nortel Networks'. *MIS Quarterly* **26**(3): pp.269-289.
- Meyerson, D., K. E. Weick and R. M. Kramer (1996). 'Swift trust and temporary groups'. *Trust in organizations: Frontiers of theory and research*. T. R. Tyler. Thousand Oaks, CA, Sage.
- Miles, M. B. and A. M. Huberman (1994). *Qualitative Data Analysis: an expanded sourcebook*, SAGE Publication.
- Mockus, A. and J. D. Herbsleb (2002). 'Expertise Browser: A Quantitative Approach to Identifying Expertise'. *International Conference on Software Engineering*, Orlando, FL, USA.
- Mockus, A. and D. M. Weiss (2001). 'Globalization by Chunking: A Quantitative Approach'. *IEEE Software*(March/April): pp.30-37.
- Murugesan, S. (1999). 'Leverage Global Software Development and Distribution Using the Internet and Web'. *Cutter IT Journal* **12**(3): pp.57-63.
- Nellore, R. and R. Balachandra (2001). 'Factors Influencing Success in Integrated Product Development (IPD) Projects'. *IEEE Transactions on Engineering Management* **48**(2): pp.164-174.
- Nelson, K. M. and J. G. Coopridge (1996). 'The Contribution of Shared Knowledge to IS Group Performance'. *MIS Quarterly* **20**(4): pp.409-432.



- Nonaka, I. and H. Takeuchi (1995). *The knowledge-creating company*, Oxford University Press.
- Olin, J. G., N. P. Greis and J. D. Kasarda (1999). 'Knowledge Management Across Multi-tier Enterprise: The Promise of Intelligent Software in the Auto Industry'. *European Management Journal* **17**(4): pp.335-347.
- Olson, J. S. and G. M. Olson (2004). 'Culture Surprises in Remote Software Development Teams'. *Queue* **1**(9): pp.52-59.
- Orlikowski, W. J. (2002). 'Knowing in Practice: Enacting a Collective Capability in Distributed Organizing'. *Organization Science* **13**(3): pp.249-273.
- Orr, J. (1990). 'Sharing Knowledge Celebrating Identity: Community Memory in a Service Culture'. *Collective Remembering*. D. M. a. D. Edwards. London, Sage.
- Paasivaara, M. (2003). 'Communication Needs, Practices and Supporting Structures in Global Inter-Organizational Software Development Projects'. *Workshop on Global Software Development, part of the International Conference on Software Engineering (ICSE)*, Portland, Oregon, USA.
- Palvia, P., E. Mao, A. F. Salam and K. S. Soliman (2003). 'Management Information System Research: What's there in a Methodology?' *Communications of the Association for Information Systems* **11**: pp.289-309.
- Patton, M. Q. (2001). *Qualitative research and evaluation methods*. Thousand Oaks, CA, Sage Publications.
- Peters, J. F. and W. Pedrycz (2000). *Software Engineering: An Engineering Approach*. New York, John Wiley & Sons, Inc.
- Pettigew, A. M. (1990). 'Longitudinal Field Research on Change: Theory and Practice'. *Organization Science* **1**(3): pp.267-292.
- Pfister, C. (1997). *Component Software: A Case Study Using BlackBox Components*. Zurich, Oberon Microsystems, Inc.
- Polanyi, M. (1967). *The Tacit Dimension*. London, Routledge.
- Rafii, F. (1995). 'How Important Is Physical Collocation to Product Development Success?' *Business Horizons*(January-February): pp.78-84.
- Ravichandran, T. and M. A. Rothenberger (2003). 'Software Reuse Strategies and Component Markets'. *Communications of the ACM* **46**(8): pp.109-114.

- Repenning, A., A. Ioannidou, M. Payton, W. Ye and J. Roschelle (2001). 'Using Components for Rapid Distributed Software Development'. *IEEE Software*(March/April): pp.38-45.
- Ryan, G. W. and H. R. Bernard (2000). 'Data Management and Analysis Methods'. *Handbook of Qualitative Research*. N. K. Denzin and Y. S. Lincoln. Thousand Oaks, CA, Sage: pp.769-802.
- Sarker, S. and S. Sahay (2003). 'Understanding Virtual Team Development: an Interpretive Study'. *Journal of the Association for Information Systems* **4**: pp.1-38.
- Sarker, S. and S. Sahay (2004). 'Implications of space and time for distributed work: an interpretive study of US-Norwegian system development teams'. *European Journal of Information Systems* **13**(1): pp.3-20.
- Scott, W. R. (1992). *Organizations: Rational, Natural, and Open Systems*. Englewood Cliffs, New Jersey, Prentice-Hall.
- Smith, P. G. and E. L. Blanck (2002). 'From Experience: Leading Dispersed Teams'. *The Journal of Product Innovation Management* **19**: pp.294-304.
- Starr, M. K. (1965). 'Modular Production; a new concept'. *Harvard Business Review* **43**(November-December): pp.131-142.
- Storck, J. (2000). 'Knowledge diffusion through "strategic communities"'. *Sloan Management Review* **41**(2): pp.63-74.
- Strauss, A. L. and J. M. Corbin (1998). *Basics of Qualitative Research*. Thousand Oaks, CA, Sage Publications.
- Szyperski, C. (1998). *Component Software Beyond Object-Oriented Programming*. New York, Addison-Wesley.
- Terwiesch, C. and C. H. Loch (1996). 'The Role of Uncertainty Reduction in Concurrent Engineering: An Analytical Model and Emperical Testing'.
- Traas, V. and J. van Hillegersberg (2000). 'The Software Component Market on the Internet: Current Status and Conditions for Growth'. *ACM SIGSOFT* **25**(1): pp.114-117.
- Trevor, J. (1994). 'RAD takes developers across the waterfall'. *Computing Canada* **20**(2): pp.22.
- Tsang, E. W. K. (1997). 'Organizational learning and the learning organization: A dichotomy between descriptive and prescriptive research'. *Human Relations* **50**(1): pp.73-89.

- Turnlund, M. (2004). 'Distributed Development Lessons Learned'. *Queue* 1(9): pp.26-31.
- Ulrich, K. T. and S. D. Eppinger (2000). *Product Design and Development*, McGraw-Hill.
- van den Heuvel, W. J., J. van Hillegersberg and M. Papazoglou (2002). 'A methodology to support Web-services Development using Legacy Systems'. *Engineering information systems in the Internet Context, IFIP 8.1 EISIC*, Kluwer Academic Publishers.
- van Fenema, P. C. (2002). 'Coordination and Control of Globally Distributed Software Projects' (Doctoral thesis). Department of Decision and Information Sciences. *Erasmus University, Rotterdam School of Management: 572*.
- van Fenema, P. C. and K. Kumar (2000). 'Coupling, Interdependence and Control in Global Projects'. *Projects as Business Constituents and Guiding Motives*. R. A. Lundin and F. Hartman. Boston, MA, Kluwer Academic Publishers.
- van Hillegersberg, J. (2003). 'Component-Based Systems Development: The AEGON Bank Project'. Executive class, Florida International University, Miami, FL, USA.
- Verbraeck, A., Y. Saanen, Z. Stojanovic, E. Valentin, K. van der Meer, A. Meijer and B. Shishkov (2002). 'What are Building Blocks?' *Building Blocks for Effective Telematics Application Development and Evaluation*, Delft University of Technology.
- Vitharana, P. (2003). 'Risks and Challenges of Component-Based Software Development'. *Communications of the ACM* 46(8): pp.67-72.
- Wallace, L. and M. Keil (2004). 'Software Project Risks and their Effect on Outcomes'. *Communications of the ACM* 47(4): pp.68-73.
- Wang, H. and C. Wang (2001). 'Open Source Software Adoption: A Status Report'. *IEEE Software* 18(2): pp.90-95.
- Webster (1992). *Webster's Dictionary*. Oxford University Press.
- Wegner, D. M. (1987). 'Transactive Memory: A Contemporary Analysis of the Group Mind'. *Theories of Group Behavior*. G. Mullen and G. Goethals. New York, Springer Verlag.

- Weick, K. E. and K. H. Roberts (1993). 'Collective Mind in Organisations: Heedful Interrelating on Flight Desks'. *Administrative Science Quarterly* **38**(3): pp.357-382.
- Weick, K. E., K. M. Sutcliffe and D. Obstfeld (1999). 'Organizing for high reliability: Processes of collective mindfulness'. *Research in organizational behavior*. B. Staw and L. L. Cummings. Greenwich, CT, JAI Press. **21**: pp.81-123.
- Weitzman, E. A. (2000). 'Software and Qualitative Research'. *Handbook of Qualitative Research*. N. K. Denzin and Y. S. Lincoln. Thousand Oaks, CA, Sage: pp.803-820.
- Willcocks, L., P. Petherbridge and N. Olson (2002). *Making IT count. Strategy, Delivery, Infrastructure*. Oxford, Butterworth Heinemann.
- Yenne, B. (2002). *Inside Boeing 777: Building The 777*. St. Paul, MBI Publishing Company.
- Yin, R. K. (1994). *Case Study Research: Design and Methods*. Newbury Park, CA, Sage.



## **CURRICULUM VITAE**

Julia Kotlarsky graduated in 1996 as an engineer in Industrial Engineering and Management from Technion, the Israeli Institute of Technology. In 1997-1999, during study for her master's degree, she received the highest level of fellowship at the Technion and worked as a teaching assistant for a number of undergraduate and graduate courses at the Department of Industrial Engineering and Management. At the same period she worked as a lecturer in ORT College.

During her graduate studies at Rotterdam School of Management, Erasmus University, The Netherlands, in 2001 she received a Marie Curie Scholarship from the European Commission for visiting Henley Management College, Henley-on-Thames, UK, where she had a visiting position for five months. In 2001-2003 she also visited Florida International University in Miami, USA, the University of Technology Sydney in Sydney, Australia, and the Management Development Institute in Gurgaon, India. She conducted research in leading companies, which included SAP, Tata Consultancy Services (TCS), Baan and LeCroy Corporation. Currently she is a lecturer in Information Systems at Warwick Business School, UK.

In 2003 she won the Philip Law Scholarship from the European Case Clearing House for writing a teaching case about globally distributed development of component-based software. She has published in International Journal of Production Research, European Journal of Information Systems and has given a number of papers in refereed conferences.

Julia is working in the area of management of globally distributed software development projects. Her interests include component-based design, knowledge sharing, and the social and technical aspects of the management of globally distributed software development projects.

**ERASMUS RESEARCH INSTITUTE OF MANAGEMENT (ERIM)**

ERIM PH.D. SERIES  
RESEARCH IN MANAGEMENT

ERIM Electronic Series Portal: <http://hdl.handle.net/1765/1>

Appelman, J.H., *Governance of Global Interorganizational Tourism Networks; Changing Forms of Co-ordination between the Travel Agency and Aviation Sector*, Promotors: Prof. dr. F.M. Go & Prof. dr. B. Nootboom, EPS-2004-036-MKT, ISBN 90-5892-060-7, <http://hdl.handle.net/1765/1199>

Berens, G., *Corporate Branding: The Development of Corporate Associations and their Influence on Stakeholder Reactions*, Promotor: Prof. dr. C. B. M. van Riel, EPS-2004-039-ORG, ISBN 90-5892-065-8, <http://hdl.handle.net/1765/1273>

Berghe, D.A.F., *Working Across Borders: Multinational Enterprises and the Internationalization of Employment*, Promotors: Prof. dr. R.J.M. van Tulder & Prof. dr. E.J.J. Schenk, EPS-2003-029-ORG, ISBN 90-5892-05-34, <http://hdl.handle.net/1765/1041>

Bijman, W.J.J., *Essays on Agricultural Co-operatives; Governance Structure in Fruit and Vegetable Chains*, Promotor: Prof. dr. G.W.J. Hendrikse, EPS-2002-015-ORG, ISBN: 90-5892-024-0, <http://hdl.handle.net/1765/867>

Brito, M.P. de, *Managing Reverse Logistics or Reversing Logistics Management?* Promotors: Prof. dr. ir. R. Dekker & Prof. dr. M. B. M. de Koster, EPS-2004-035-LIS, ISBN: 90-5892-058-5, <http://hdl.handle.net/1765/1132>

Campbell, R.A.J., *Rethinking Risk in International Financial Markets*, Promotor: Prof. dr. C.G. Koedijk, EPS-2001-005-F&A, ISBN: 90-5892-008-9, <http://hdl.handle.net/1765/306>

Chen, Y., *Labour Flexibility in China's Companies: An Empirical Study*, Promotors: Prof. dr. A. Buitendam & Prof. dr. B. Krug, EPS-2001-006-ORG, ISBN: 90-5892-012-7, <http://hdl.handle.net/1765/307>

Daniševská, P., *Empirical Studies on Financial Intermediation and Corporate Policies*, Promotor: Prof. dr. C.G. Koedijk, EPS-2004-044-F&A, ISBN 90-5892-070-4, <http://hdl.handle.net/1765/1518>

Delporte-Vermeiren, D.J.E., *Improving the Flexibility and Profitability of ICT-enabled Business Networks: An Assessment Method and Tool*, Promotors: Prof. mr. dr. P.H.M. Vervest & Prof. dr. ir. H.W.G.M. van Heck, EPS-2003-020-LIS, ISBN: 90-5892-040-2, <http://hdl.handle.net/1765/359>

Dijksterhuis, M., *Organizational Dynamics of Cognition and Action in the Changing Dutch and US Banking Industries*, Promotors: Prof. dr. ing. F.A.J. van den Bosch & Prof. dr. H.W. Volberda, EPS-2003-026-STR, ISBN: 90-5892-048-8, <http://hdl.handle.net/1765/1037>

Fenema, P.C. van, *Coordination and Control of Globally Distributed Software Projects*, Promotor: Prof. dr. K. Kumar, EPS-2002-019-LIS, ISBN: 90-5892-030-5, <http://hdl.handle.net/1765/360>

Fleischmann, M., *Quantitative Models for Reverse Logistics*, Promoters: Prof. dr. ir. J.A.E.E. van Nunen & Prof. dr. ir. R. Dekker, EPS-2000-002-LIS, ISBN: 3540 417 117, <http://hdl.handle.net/1765/1044>



Flier, B., *Strategic Renewal of European Financial Incumbents: Coevolution of Environmental Selection, Institutional Effects, and Managerial Intentionality*, Promotors: Prof. dr. ing. F.A.J. van den Bosch & Prof. dr. H.W. Volberda, EPS-2003-033-STR, ISBN: 90-5892-055-0, <http://hdl.handle.net/1765/1071>

Fok, D., *Advanced Econometric Marketing Models*, Promotor: Prof. dr. P.H.B.F. Franses, EPS-2003-027-MKT, ISBN: 90-5892-049-6, <http://hdl.handle.net/1765/1035>

Ganzaroli, A., *Creating Trust between Local and Global Systems*, Promotors: Prof. dr. K. Kumar & Prof. dr. R.M. Lee, EPS-2002-018-LIS, ISBN: 90-5892-031-3, <http://hdl.handle.net/1765/361>

Gilsing, V.A., *Exploration, Exploitation and Co-evolution in Innovation Networks*, Promotors: Prof. dr. B. Nooteboom & Prof. dr. J.P.M. Groenewegen, EPS-2003-032-ORG, ISBN 90-5892-05-42, <http://hdl.handle.net/1765/1040>

Graaf, G. de, *Tractable Morality: Customer Discourses of Bankers, Veterinarians and Charity Workers*, Promotors: Prof. dr. F. Leijnse & Prof. dr. T. van Willigenburg, EPS-2003-031-ORG, ISBN: 90-5892-051-8, <http://hdl.handle.net/1765/1038>

Hermans, J.M., *ICT in Information Services, Use and deployment of the Dutch securities trade, 1860-1970*. Promotor: Prof. dr. drs. F.H.A. Janszen, EPS-2004-046-ORG, ISBN 90-5892-072-0, <http://hdl.handle.net/1765/1793>

Heugens, P.M.A.R., *Strategic Issues Management: Implications for Corporate Performance*, Promotors: Prof. dr. ing. F.A.J. van den Bosch & Prof. dr. C.B.M. van Riel, EPS-2001-007-STR, ISBN: 90-5892-009-7, <http://hdl.handle.net/1765/358>

Hooghiemstra, R., *The Construction of Reality*, Promotors: Prof. dr. L.G. van der Tas RA & Prof. dr. A.Th.H. Pruyn, EPS-2003-025-F&A, ISBN: 90-5892-047-X, <http://hdl.handle.net/1765/871>

Jansen, J.J.P., *Ambidextrous Organizations*, Promotors: Prof.dr.ing. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2005-055-STR, ISBN 90-5892-081-X

Jong, C. de, *Dealing with Derivatives: Studies on the Role, Informational Content and Pricing of Financial Derivatives*, Promotor: Prof. dr. C.G. Koedijk, EPS-2003-023-F&A, ISBN: 90-5892-043-7, <http://hdl.handle.net/1765/1043>

Kippers, J., *Empirical Studies on Cash Payments*, Promotor: Prof.dr. Ph.H.B.F. Franses, EPS-2004-043-F&A. ISBN 90-5892-069-0, <http://hdl.handle.net/1765/1520>

Koppius, O.R., *Information Architecture and Electronic Market Performance*, Promotors: Prof. dr. P.H.M. Vervest & Prof. dr. ir. H.W.G.M. van Heck, EPS-2002-013-LIS, ISBN: 90-5892-023-2, <http://hdl.handle.net/1765/921>

Langen, P.W. de, *The Performance of Seaport Clusters; A Framework to Analyze Cluster Performance and an Application to the Seaport Clusters of Durban, Rotterdam and the Lower Mississippi*, Promotors: Prof. dr. B. Nooteboom & Prof. drs. H.W.H. Welters, EPS-2004-034-LIS, ISBN: 90-5892-056-9, <http://hdl.handle.net/1765/1133>

Le Anh, T., *Intelligent Control of Vehicle-Based Internal Transport Systems*, Promotors: Prof.dr. M.B.M. de Koster & Prof.dr.ir. R. Dekker, EPS-2005-051-LIS, ISBN 90-5892-079-8

Liang, G., *New Competition; Foreign Direct Investment And Industrial Development In China*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2004-047-ORG, ISBN 90-5892-073-9, <http://hdl.handle.net/1765/1795>.

Loef, J., *Incongruity between Ads and Consumer Expectations of Advertising*, Promotors: Prof. dr. W.F. van Raaij & Prof. dr. G. Antonides, EPS-2002-017-MKT, ISBN: 90-5892-028-3, <http://hdl.handle.net/1765/869>

Mandele, L.M., van der, *Leadership and the Inflection Point: A Longitudinal Perspective*, Promotors: Prof. dr. H.W. Volberda, Prof. dr. H.R. Commandeur, EPS-2004-042-STR, ISBN 90-5892-067-4, <http://hdl.handle.net/1765/1302>

Meer, J.R. van der, *Operational Control of Internal Transport*, Promotors: Prof. dr. M.B.M. de Koster & Prof. dr. ir. R. Dekker, EPS-2000-001-LIS, ISBN:90-5892-004-6, <http://hdl.handle.net/1765/859>

Miltenburg, P.R., *Effects of Modular Sourcing on Manufacturing Flexibility in the Automotive Industry: A Study among German OEMs*, Promotors: Prof. dr. J. Paauwe & Prof. dr. H.R. Commandeur, EPS-2003-030-ORG, ISBN 90-5892-052-6, <http://hdl.handle.net/1765/1039>

Mol, M.M., *Outsourcing, Supplier-relations and Internationalisation: Global Source Strategy as a Chinese Puzzle*, Promotor: Prof. dr. R.J.M. van Tulder, EPS-2001-010-ORG, ISBN: 90-5892- 014-3, <http://hdl.handle.net/1765/355>

Mulder, A., *Government Dilemmas in the Private Provision of Public Goods*, Promotor: Prof. dr. R.J.M. van Tulder, EPS-2004-045-ORG, ISBN: 90-5892-071-2, <http://hdl.handle.net/1765>

Muller, A.R., *The Rise of Regionalism: Core Company Strategies Under The Second Wave of Integration*, Promotor: Prof. dr. R.J.M. van Tulder, EPS-2004-038-ORG, ISBN 90-5892-062-3, <http://hdl.handle.net/1765/1272>

Oosterhout, J. van, *The Quest for Legitimacy: On Authority and Responsibility in Governance*, Promotors: Prof. dr. T. van Willigenburg & Prof. mr. H.R. van Gunsteren, EPS-2002-012-ORG, ISBN: 90-5892-022-4, <http://hdl.handle.net/1765/362>

Peeters, L.W.P., *Cyclic Railway Timetable Optimization*, Promotors: Prof. dr. L.G. Kroon & Prof. dr. ir. J.A.E.E. van Nunen, EPS-2003-022-LIS, ISBN: 90-5892-042-9, <http://hdl.handle.net/1765/429>

Popova, V., *Knowledge Discovery and Monotonicity*, Promotor: Prof. dr. A. de Bruin, EPS-2004-037-LIS, ISBN 90-5892-061-5, <http://hdl.handle.net/1765/1201>

Puvasasvari Ratnasingam, P., *Interorganizational Trust in Business to Business E-Commerce*, Promotors: Prof. dr. K. Kumar & Prof. dr. H.G. van Dissel, EPS-2001-009-LIS, ISBN: 90-5892-017-8, <http://hdl.handle.net/1765/356>

Romero Morales, D., *Optimization Problems in Supply Chain Management*, Promotors: Prof. dr. ir. J.A.E.E. van Nunen & Dr. H.E. Romeijn, EPS-2000-003-LIS, ISBN: 90-9014078-6, <http://hdl.handle.net/1765/865>

Roodbergen, K.J., *Layout and Routing Methods for Warehouses*, Promotors: Prof. dr. M.B.M. de Koster & Prof. dr. ir. J.A.E.E. van Nunen, EPS-2001-004-LIS, ISBN: 90-5892-005-4, <http://hdl.handle.net/1765/861>

Schweizer, T.S., *An Individual Psychology of Novelty-Seeking, Creativity and Innovation*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2004-048-ORG, ISBN: 90-5892-07-71, <http://hdl.handle.net/1765/1818>

Six, F.E., *Trust and Trouble: Building Interpersonal Trust Within Organizations*, Promotors: Prof. dr. B. Nooteboom & Prof. dr. A.M. Sorge, EPS-2004-040-ORG, ISBN 90-5892-064-X, <http://hdl.handle.net/1765/1271>

Slager, A.M.H., *Banking across Borders*, Promotors: Prof. dr. D.M.N. van Wensveen & Prof. dr. R.J.M. van Tulder, EPS-2004-041-ORG, ISBN 90-5892-066-6, <http://hdl.handle.net/1765/1301>

Speklé, R.F., *Beyond Generics: A closer look at Hybrid and Hierarchical Governance*, Promotor: Prof. dr. M.A. van Hoepen RA, EPS-2001-008-F&A, ISBN: 90-5892-011-9, <http://hdl.handle.net/1765/357>

Teunter, L.H., *Analysis of Sales Promotion Effects on Household Purchase Behavior*, Promotors: Prof. dr. ir. B. Wierenga & Prof. dr. T. Kloek, EPS-2002-016-ORG, ISBN: 90-5892-029-1, <http://hdl.handle.net/1765/868>

Valck, K. de, *Virtual Communities of Consumption: Networks of Consumer Knowledge and Companionship*, Promotors: Prof.dr.ir. G.H. van Bruggen, & Prof.dr.ir. B. Wierenga, EPS-2005-050-MKT, ISBN 90-5892-078-X

Verheul, I., *Is there a (fe)male approach? Understanding gender differences in entrepreneurship*, Prof.dr. A.R. Thurik, EPS-2005-054-ORG, ISBN 90-5892-080-1

Vis, I.F.A., *Planning and Control Concepts for Material Handling Systems*, Promotors: Prof. dr. M.B.M. de Koster & Prof. dr. ir. R. Dekker, EPS-2002-014-LIS, ISBN: 90-5892-021-6, <http://hdl.handle.net/1765/866>

Vliet, P. van, *Downside Risk and Empirical Asset Pricing*, Promotor: Prof. dr. G.T. Post, EPS-2004-049-F&A ISBN 90-5892-07-55, <http://hdl.handle.net/1765/1819>

Waal, T. de, *Processing of Erroneous and Unsafe Data*, Promotor: Prof. dr. ir. R. Dekker, EPS-2003-024-LIS, ISBN: 90-5892-045-3, <http://hdl.handle.net/1765/870>

Wielemaker, M.W., *Managing Initiatives: A Synthesis of the Conditioning and Knowledge-Creating View*, Promotors: Prof. dr. H.W. Volberda & Prof. dr. C.W.F. Baden-Fuller, EPS-2003-28-STR, ISBN 90-5892-050-X, <http://hdl.handle.net/1765/1036>

Wijk, R.A.J.L. van, *Organizing Knowledge in Internal Networks: A Multilevel Study*, Promotor: Prof. dr. ing. F.A.J. van den Bosch, EPS-2003-021-STR, ISBN: 90-5892-039-9, <http://hdl.handle.net/1765/347>

Wolters, M.J.J., *The Business of Modularity and the Modularity of Business*, Promotors: Prof. mr. dr. P.H.M. Vervest & Prof. dr. ir. H.W.G.M. van Heck, EPS-2002-011-LIS, ISBN: 90-5892-020-8, <http://hdl.handle.net/1765/920>

## Management of Globally Distributed Component-Based Software Development Projects

Globally Distributed Component-Based Development (GD CBD) is expected to become a promising area, as increasing numbers of companies are setting up software development in a globally distributed environment and at the same time are adopting CBD methodologies. Being an emerging area, the management of GD CBD has evolved primarily on an ad hoc basis. At present, little is known about how to successfully organise and manage GD CBD. To fill this gap, this research explores the management of GD CBD and reveals factors that contribute to success in GD CBD projects. Data are drawn from several successful GD CBD projects at LeCroy, SAP and TCS, compared with one unsuccessful project at Baan. The results suggest that inter-site coordination, appropriate tools and technologies, social ties, knowledge sharing and components management are the main factors that contribute to success in GD CBD. Lastly, a framework assisting managers to organize and manage CBD in GD environments is offered.

### ERIM

The Erasmus Research Institute of Management (ERIM) is the Research School (Onderzoekschool) in the field of management of the Erasmus University Rotterdam. The founding participants of ERIM are RSM Erasmus University and the Erasmus School of Economics. ERIM was founded in 1999 and is officially accredited by the Royal Netherlands Academy of Arts and Sciences (KNAW). The research undertaken by ERIM is focussed on the management of the firm in its environment, its intra- and inter-firm relations, and its business processes in their interdependent connections.

The objective of ERIM is to carry out first rate research in management, and to offer an advanced graduate program in Research in Management. Within ERIM, over two hundred senior researchers and Ph.D. candidates are active in the different research programs. From a variety of academic backgrounds and expertises, the ERIM community is united in striving for excellence and working at the forefront of creating new business knowledge.