Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

**Nico Hiort af Ornäs**

# Preparing and Initiating a Globally Distributed Software Development Project

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, June 15, 2010

|  |  |
|---|---|
| Supervisor: | Professor Casper Lassenius |
| Instructor: | Ph.D. Maria Paasivaara |

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF THE
MASTER'S THESIS

Author: Nico Hiort af Ornäs

Title: Preparing and Initiating a Globally Distributed Software Development Project

Date: 15.6.2010 Language: English Number of pages:9+91

Faculty of Information and Natural Sciences

Department of Computer Science and Engineering

Professorship: Computer science Code: T-76

Supervisor: Prof. Casper Lassenius

Instructor: D.Sc. (Tech.) Maria Paasivaara

As globalization is driving organizations to become more and more distributed, multi-site development is becoming a norm. With the increasing globalization in this industry, it is necessary to better prepare software development projects to manage work in distributed environments.
Several difficulties exist in preparing and initiating a globally distributed project. As most distributed projects underestimate the required time and resources needed for a successful project ramp-up, the projects tend to be initiated with a lack of proper planning and preparation. Thus, this work attempts to address how the early-life of globally distributed software development projects should be managed, with an aim of finding successful practices for preparing and initiating distributed projects.
The research approach for this study is an exploratory multiple case study consisting of four case projects, with all case projects being within the boundaries of a single case organization. The data collection method used for these case projects was semi-structured interviews.
The results of this study consist of a set of successful practices and recommendations for preparing and initiating global software development projects. Preparation is required regarding different approaches for structuring the teams and the organization, and establishing a plan on what to distribute. A successful project initiation requires a face-to-face kick-off meeting, training of new team members, knowledge transfer regarding the product and business domain, and designation of liaisons.

Keywords: global, distributed, software, development, preparation, initiation

AALTO-YLIOPISTO
TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Nico Hiort af Ornäs

Työn nimi: Maailmanlaajuisesti hajautettujen ohjelmistoprojektien valmistelu ja käynnistäminen

Päivämäärä: 15.6.2010 Kieli: Englanti Sivumäärä:9+91

Informaatio- ja luonnontieteiden tiedekunta

Tietotekniikan laitos

Professuuri: Tietojenkäsittelyoppi Koodi: T-76

Valvoja: Prof. Casper Lassenius

Ohjaaja: TkT Maria Paasivaara

Organisaatioiden välinen kansainvälinen yhteistyö on tekemässä tämän päivän ohjelmistokehityksestä yhä enemmän hajautettua. Koska globalisaatio on kasvamassa IT-alalla, tulee maailmanlaajuisesti hajautettujen ohjelmistoprojektien valmistelu ja käynnistäminen hoitaa paremmin, jotta töiden hallinta onnistuisi hajautetussa ympäristössä.
Maailmanlaajuisten ohjelmistokehityprojektien valmistautumiseen ja käynnistämiseen liittyy useita eri haasteita ja ongelmia. Hajautetut ohjelmistokehitysprojektit aliarvioivat yleensä alun merkitystä projektin menestykselle, jonka takia projekteja yleensä käynnistetään ilman kunnon suunnittelua ja valmistelua. Näin ollen tässä työssä on pyritty käsittelemään kuinka kansainvälisten ohjelmistoprojektien alkua tulisi hallinnoida. Yhtenä työn tavoitteena on ollut löytää onnistuneita käytäntöjä hajautettujen projektien valmisteluun ja käynnistämiseen.
Tämä tutkimus on luonteeltaan eksploratiivinen, jossa aineisto on kerätty yhdestä organisaatiosta haastattelemalla projektien jäseniä neljästä eri projektista. Aineiston keräystä varten käytettiin puolistrukturoituja haastatteluja.
Tämän tutkimuksen tulokset koostuvat onnistuneista käytännöistä ja suosituksista, joita voidaan käyttää hajautettujen projektien valmisteluun ja käynnistämiseen. Projektin valmistelu koostuu muun muassa tiimi- ja organisaatiorakenteen päättämisestä sekä hajautuksen suunnittelusta. Onnistunut projektin käynnistäminen edellyttää aloituskokouksen pitämistä, joka tulisi pitää kasvokkain kaikkien projektin jäsenien kanssa, uusien tiimijäsenten kouluttamista sekä yhteyshenkilöiden nimittämistä.

Avainsanat: maailmanlaajuinen, hajautettu, ohjelmistokehitys, ohjelmistotuotanto, valmistelu, käynnistäminen

# Acknowledgements

This Master's Thesis has been written at SoberIT from September 2009 till June 2010. I want to give many thanks to everybody who have helped me during this time.

I want to thank my supervisor Professor Casper Lassenius for giving me valuable feedback and steering me to the right direction. I would like to give special thanks to Ph.D. Maria Paasivaara for her continuous support and invaluable feedback. She has reviewed this thesis several times which has greatly contributed to this work.

I am grateful to SoberIT for giving me the opportunity to finish my Master's Thesis. During my time at SoberIT, I have had the pleasure to work with many colleagues whose ideas and feedback have greatly improved the outcome of this thesis. I would especially like to thank the whole MaPIT team for their support and for collecting the data for this study. I am also thankful of all the humor and joy that Peitsa Hynninen brought into my working days.

I would also like to thank my family and friends for their support, especially my mother Pirkko and my father Kim. I also want to thank my girlfriend Hu Dan for the love, support and understanding she has given me.

Espoo, June 15, 2010

Nico Hiort af Ornäs

# Contents

# Abbreviations

| | |
|---|---|
| GSD | Global Software Development |
| RUP | Rational Unified Process |

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This chapter presents a short introduction of this thesis. The chapter begins with section 1.1 giving a short introduction of global software development and a motivation of why the chosen topic is important. Section 1.2 presents the aims and objectives of this study, and shortly introduces the research problem. Section 1.3 presents the scope of this thesis and the organizational context in which this study has been conducted. Finally, the chapter concludes with a thesis outline in section 1.4.

## 1.1 Background and motivation

Global software development (GSD), either through inter-organizational collaboration or outsourcing, is becoming a common model that can be seen in many software projects today. Locating development teams in different countries is no longer considered as unusual. Many organizations are engaging in GSD to benefit from decreased costs and a larger pool of labor. Globalization of software projects is further driven by the availability of technically skilled resources around the globe. (Carmel, 1999)

Given these opportunities, there are also several difficulties in initiating a globally distributed project. The main challenges are related to the difficulty of maintaining efficient communication, coordination and control when teams are distributed across sites (Carmel & Agarwal, 2001). Most distributed software development projects underestimate the required time and resources needed for a successful project ramp-up (Jensen *et al.* , 2007). Moreover, the team members of a distributed project are faced with many challenges in the beginning of a project. Bringing new employees up to speed can take a considerable amount of time and effort, as they have to learn to understand the business, products, processes and the culture of the contracting organization (Herbsleb *et al.* , 2005). In addition, building trust and a proper culture for a distributed environment is also difficult, both of which are needed for a long-lasting cooperation to occur between the sites (Hofner *et al.* , 2007). While many studies have touched on the various problems in globally distributed software projects, there is a lack of qualitative studies which concentrate explicitly on the preparation and ramp-up phases of distributed software projects. Moreover, the existence of these various problems in the ramp-up phase of a distributed project makes this an attractive project phase to study.

Global software development can however be a costly endeavor, with training of new team members and traveling across sites requiring a large investment (Kobitzsch *et al.* , 2001). As the cost of starting a globally distributed project can be relatively high, there is an increasing interest for organizations to learn how to successfully start a distributed software project. In addition, as the benefits of decreased costs can only be realized in the long-term view, it is essential for organizations to be able to quickly ramp-up their projects in order to achieve cost savings (Khan *et al.* , 2009). Because of this same reason, the ramp-up phase of a distributed project does often not get enough attention as projects tend to be initiated with a lack of proper planning and preparation. As such, it is important to study the preparation and ramp-up phases of distributed projects in order to understand how these projects can be started successfully so that the long-term benefits can eventually be realized.

## 1.2 Aims and objectives

This thesis aims to study how work should be managed in the early-life of a distributed software development project. Thus, an exploratory research approach was taken to conduct interviews in four case projects by collecting data through semi-structured interviews. The outcome of this study is a set of recommendations of different practices, approaches and methods that should be considered when planning and initiating a distributed software development project.

The main aim of this study is to discover helpful practices, methods and techniques that could be used in the beginning of a distributed software development project in order for the project to have a chance of success. These range from establishing a plan on what to distribute, structuring the teams and the organization, providing training and knowledge transfer to team members, establishing efficient working practices, to designating liaisons. As such, the research problem for this study has been formed as follows:

*What should be done in the early-life of a distributed software development project in order for it to have a better chance of success?*

This research problem is further divided into separate research questions in section 2.1.

## 1.3 Scope

This study is limited to four case projects, referred to as Case Alpha, Case Beta, Case Gamma and Case Delta. All the studied case projects consisted of one main site (on-site) and one remote site (off-site), except for Case Delta which had several remote sites offering maintenance and customer support activities. The main site was usually involved in business management and product analysis, whereas the remote site was often doing development and testing. All the studied case projects belonged to the same case organization that performs offshore development on a global scale. This case organization will not be named in this thesis due to reasons related to confidentiality. Whereas all the involved case projects are in-house projects, the conducted literature study in chapter 3 will also focus on other organizational contexts, such as outsourcing with an external subcontractor. This is further clarified in figure 1.1.

The scope of this thesis is limited to the field of software development, although

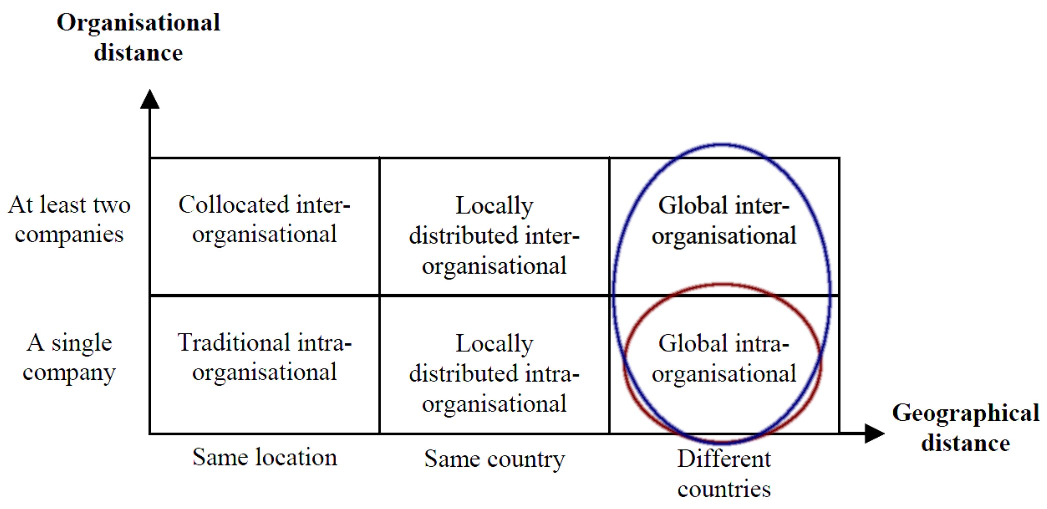
Organisational distance
At least two companies
A single company
Collocated inter-organisational
Locally distributed inter-organisational
Global inter-organisational
Traditional intra-organisational
Locally distributed intra-organisational
Global intra-organisational
Same location
Same country
Different countries
Geographical distance

Figure 1.1: A diagram (modified from Paasivaara (2005)) showing the organizational context in which this study has been conducted. The larger blue circle indicates the focus of the literature study, whereas the smaller red circle indicates the focus of the analyzed case projects.

some of the findings may as well be applicable for other types of distributed projects where knowledge-intensive work is required. In all of the studied case projects, the work consisted of development of new software, or of maintenance work where new features were developed, both of which tend to be categorized as knowledge-intensive.

This study is further limited to the early-life of distributed software development projects. This includes project-related work that takes place both before and after the project is initiated. However, the selection of partners and subcontracting organizations is outside the scope of this thesis. Moreover, one requirement for all the studied case projects was that both sites were involved in software development activities, making tight collaboration and communication between the sites a requirement as well. This was very desirable, as it allowed the effects and difficulties of distribution to be studied.

## 1.4 Thesis outline

Chapter 2 presents the research approach and method used in this study. The chapter also shortly presents the case projects and describes how they were selected to be included in this study. The approach used to conduct the literature study is also described in this chapter.

Chapter 3 introduces previous work related to this study.

Chapter 4 presents the results that were obtained from the studied case projects. First, the chapter continues to describe the studied case projects more elaborately. After this, the chapter presents practices and approaches that should be considered before and after initiating a distributed software project.

Chapter 5 answers to the research questions and makes final conclusions based on the findings from the case projects and literature.

# Chapter 2

# Research Methodology

This chapter presents the research methods used in this study. First, the chapter begins with stating the research problem and questions in section 2.1. Next, the research approach used to solve the research problem and questions is presented in section 2.2. Section 2.3 presents how the literature study was conducted. Section 2.4 presents the criteria for selecting the case projects for this study. After this, section 2.5 continues with a short introduction to the selected case projects. Finally, section 2.6 describes how the data was collected for this study using semi-structured interviews, and the last section 2.7 describes how the data was analyzed.

## 2.1 Research questions

The research problem this study attempts to address is formed as follows:

*What should be done in the early-life of a distributed software development project in order for it to have a better chance of success?*

The main research problem is further divided into two research questions:

- **Research Question 1**: *What factors should be taken into account when preparing for a distributed software development project?*
- **Research Question 2**: *What practices should be implemented when initiating a distributed software development project?*

## 2.2 Research approach

This study extends previous research in the field of global software development. The research approach for this study is an exploratory multiple case study (Yin, 1994), as the type of the presented research questions in section 2.1 calls for an exploratory case study analysis. According to Yin (1994), research questions focusing mainly on ”what” questions are justifiable for conducting exploratory case studies. The term exploratory means that the data collection for the study has been done before the formation of the research problem and research questions (Tellis, 1997). For this study, the research problem was defined after the collection of data.

The unit of analysis in this study is one case project. The number of cases included in this study is four, with all cases belonging to the same case organization. Because this study was conducted within the boundaries of a single case organization, there may some threats regarding generalizability. Even though the size of the organization and its global presence does create a multi-organizational diversity, the findings may still not be applicable in all organizational contexts.

## 2.3 Literature study approach

The literature study was conducted after the case studies had been analyzed. The researcher reviewed literature on global software development, mainly focusing on case studies in distributed development. The intention of the literature study was to identify relevant practices that can be initiated in the beginning of a distributed project in order for it to have a better chance of success.

The literature study was conducted using three sources in order to find relevant journal and conference articles in the field of global software development. The first source used by the researcher was the International Conference on Global Software Engineering. From all the four instances of this conference, the researcher found 25 relevant articles by inspecting the title and abstract of each article. After this, the researcher continued to search for relevant articles from two other sources, namely Google Scholar [1] and NELLI Portal [2], both of which are websites that are able to connect and find articles from various sources on the Internet. This search resulted in 34 relevant articles by inspecting the title and abstract of each article. After a more careful inspection of all the found articles, another nine articles were identified as irrelevant. This resulted in a total of 50 articles that were included in the literature study.

## 2.4 Case selection

The cases for this study were selected from one single case organization. This case organization is a participant in the MaPIT (Management, Processes and IT Support in Globally Distributed Software Development) research project at SoberIT (Software Business and Engineering Institute) of Aalto University School of Science and Technology. The MaPIT research team interviewed a total number of seven case projects from this case organization. These interviews were done in 2007. The cases were selected by MaPIT researchers together with representatives from the case organization. The purpose was to get a selection of information rich cases. For this study, the selection of cases was purposefully done to consist of four cases, two successful and two challenged cases.

As all the case projects were selected from the same case organization, it allowed most of them to share some similarities. For example, three of the cases were outsourced from the same country to the same off-site location. Because of this, one case was chosen because of its differing on-site and off-site location, in order to see if the results would hold up in a more broad range of conditions. Moreover, all of the cases involved different people at both on-site and off-site, with each case developing a different type of software. Given these differences, the case projects tended to have different working practices,

[1]Google Scholar is a free web-based search engine that indexes content from various scientific sources. URL: http://scholar.google.com/

[2]NELLI (**N**ational **EL**ectronic **L**ibrary **I**nterface) Portal is a Finnish website for managing and searching scientific information and articles from several different databases. NELLI provides open access to some resources, but most of the resources are restricted to schools and universities in Finland. URL: http://www.nelliportaali.fi/

Table 2.1: The studied case projects

| | Domain | On-site | Off-site | Team size | Interviews |
|---|---|---|---|---|---|
| Case Alpha | Communication system | Finland | Czech Republic | 6 + 26 | 2 + 4 |
| Case Beta | Financing | Finland | Czech Republic | 20 + 11 | 6 + 4 |
| Case Gamma | Enterprise resource management | Finland | Czech Republic | 17 + 19 | 5 + 4 |
| Case Delta | Enterprise resource management | Norway | Malaysia | 20 + 20 | 4 + 3 |

different set of tools, and project members with rather different skills and experiences.

The selection of four cases for this study was done by two members from the MaPIT team, as they were aware which cases were the most critical ones to be suitable for this study. The final selection of cases can be seen from table 2.1. Team size and interviews columns define the number of people at on-site and off-site respectively.

## 2.5 Case projects overview

All the case projects in this study were selected from the same case organization. This case organization had one software center located in Czech Republic, a place where three of the studied case projects, namely cases Alpha, Beta and Gamma, had located their off-site teams. In each of these cases, the on-site teams were located in one of the many offices the organization had in Finland. Case Delta differed notably from the other cases by having its on-site teams located in Norway and off-site teams in Malaysia. The case projects are summarized in table 2.1.

## 2.6 Data collection

The data for this study was collected between 2007-2008 from four case projects. The data collection method used for these case projects was semi-structured interviews. The author of this thesis did not participate in the interviews. Instead, the interviews were conducted by four researchers from the MaPIT team. With a few exceptions, two researchers participated in each interview. One of the interviewers led the conversation by asking questions, whereas the other interviewer took notes and asked additional questions. One of the interviews was conducted by three researchers because of the availability of one extra researcher. In addition, two interviews were conducted as voice calls over the Internet because of the long distance to the off-site location. These interviews were conducted by one researcher who led the conversation by asking questions, but did not take notes during the interview. All other interviews were conducted as face-to-face.

The planned duration of each interview was two hours. In reality, the interviews lasted from 1 hour to 2,5 hours. Those interviews that were conducted last within a project tended to be shorter than the earlier ones, as most project background information had already been gathered during previous interviews. The total number of conducted interviews was 32. The number of project members that were interviewed from each case project can be seen from table 2.1. With two exceptions, each project member was only interview once. In these two exceptions, the researchers decided to arrange second interviews, as the planned duration of two hours was not enough to gather all the experiences of the interviewee.

As the case projects consisted of many project members, it was unsuitable to interview all of them. As such, only a selection of project members was interviewed. The selection of interviewees was done by MaPIT researchers together with the project manager or equivalent person from each case project. This selection was purposefully done to consist of project members from both sites, and with varying roles, such as developers, testers, architects, project managers and project owners.

Two different interview templates of questions were used in the interviews. The interview templates contained open-ended questions on different topics related to software engineering. The template used in Case Alpha, Case Beta and Case Gamma, and can be found from section A.1. A slightly modified version of the template that was used in Case Delta can be found from section A.2. Both interview templates contained questions on topics such as communication and collaboration practices, used process models, monitoring practices, and knowledge transfer practices. The interviewees were asked to share and tell in their own words about their individual experiences in the project. This included both successful and less successful practices, ideas for improvement and any lessons learnt throughout the project. The author of this thesis did not participate in writing the interview templates. Both interview templates were written by MaPIT researchers.

## 2.7 Data analysis

After the interviews were conducted, the recordings of each interview were transcribed by an external party. In order for the author to first gain a general knowledge about the case projects and their background, the author began by listening to the first few interviews held of each case project. After this first introduction round, an in-depth analysis of each interview was made by the author. The analysis for an interview was done by simultaneously listening to the recording, reading the transcription, and writing notes. The notes were written using a template made by the author. This template can be found from appendix B. Using this template, the author wrote notes of each interview and subtracted quotes from the transcription for each bullet in the template. Later on, the template was expanded to include more themes related to each specific case project. After each interview had been analyzed, the author continued to combine all the notes of a case project to make a final case description of each case project. A wiki was used to store and write all notes.

# Chapter 3

# Related Work

This chapter presents the literature review. The chapter begins with an introductional section 3.1 that explains the organizational context in which the findings from the literature are applicable in. Section 3.2 presents some factors that should be considered when preparing for a distributed project. After this, the chapter continues with section 3.3 that presents some useful practices and solutions that should be implemented in the beginning of a distributed project in order for it to have a better chance of success. Finally, section 3.4 presents a summary of the findings from the literature.

## 3.1 Introduction

Distributed projects can take place in many different organizational contexts and environments. Whereas the involved case projects in this study focus solely on in-house projects where all the offshoring happens within the boundaries of the same organization, the presented practices and solutions in this literature study focus also on other contexts, such as outsourcing with an external subcontractor. This is further clarified in section 1.3. In all of the contexts, the terms on-site and client organization refer to the main site where the customer is located, whereas terms such as off-site and off-site organization refer to the remote site which is located at a low-cost country.

## 3.2 Preparing for a distributed project

### 3.2.1 Developing and communicating a GSD strategy

A strategy for GSD should be formed in the beginning of a distributed project. It is especially important to agree and communicate about the project's goals, targets and objectives, and to ensure that all commitments are clearly understood by all involved sites (Lings *et al.* , 2007). As with any project, agreements should be made in the beginning of a distributed project in order to ensure that commitments exist in written and controlled form (Ebert *et al.* , 2001). Both sites should communicate about their own strategies in order to become aware of each others's strong and weak sides. The developed strategy should try to create a win-win situation for everybody to assure their motivation during the project. (Mettovaara *et al.* , 2006) In addition, the literature widely mentions about the importance of taking a long-term strategic view (Stark *et al.* , 2006; Khan *et al.* , 2009; Tervonen & Mustonen, 2009), as a continuous cooperation will provide both parties increased benefits as time passes by.

Several strategic decisions have to be made in the beginning of a distributed project. These include decisions regarding the selection of development sites and artifacts to distribute, the product architecture, and the used task allocation strategy (Raffo & Setamanit, 2005). Most organizations develop their strategies at the project level, whereas the more successful firms consider their collaboration strategies from a organization-wide perspective (Forbath *et al.* , 2008).

The developed strategy does not only have to be communicated to the off-site organization. It is especially important to communicate about the intent and perceived benefits of GSD to the existing on-site team, as many might fear the possibility of being relocated or their jobs being threatened (Herbsleb & Moitra, 2001). As such, a discussion with on-site team members regarding their career paths should take place (Cusick & Prasad, 2006).

### 3.2.2 Early planning and preparation

The importance of advance planning and preparation in the beginning of a distributed project is widely emphasized in the literature (Cusick & Prasad, 2006; Prikladnicki *et al.* , 2003; Mettovaara *et al.* , 2006). A planning phase allows the project to get a organized and managed ramp-up (Prikladnicki *et al.* , 2003). The planning session gives a good way for the team members to engage in the project (Prikladnicki *et al.* , 2003), and allows a team and project vision to be built collaboratively (Bhat *et al.* , 2006).

Before a project can be initiated, there must be a clear statement of goals that need to be achieved during the project. All involved parties should participate in setting the project's targets and goals. It may very well be that the initial goals may not be suitable for all parties, and may thus need to be altered. Revealing differences in interests between the involved parties should be done as early as possible. (Erenkrantz & Taylor, 2003) Moreover, the overall performance in the project may suffer when the involved parties are not aligned to the same objective (Bhat *et al.* , 2006).

The first step of the planning phase should focus on what work should be distributed and why (Tervonen & Mustonen, 2009). This first step should be followed by a detailed roadmap, which defines how the project will grow in terms of team size and responsibilities. The plan should consider the additional recruits, infrastructure and other resources that will be needed in the future. It is especially important to take a long-term view of how the cooperation will evolve. (Hofner *et al.* , 2007) The product's features should be broken down into increments in order to establish a well defined roadmap for the software to be developed. This roadmap should further specify which teams are involved and what they will be doing in each location. This is especially important to ensure a reliable allocation of resources. (Ebert *et al.* , 2001) When developing the schedule and roadmap, it is important to take into account the local holidays and time zones of each site (Lings *et al.* , 2006). In addition, there are several issues that need to be dealt with in the early planning phase of the project, such as setting up supporting infrastructure and clear expectations for deliverables (Cusick & Prasad, 2006), and determining common rules, responsibilities and tools to be used (Komi-Sirviö & Tihinen, 2005).

### 3.2.3 Structuring the teams and the organization

The implementation of an effective team and organization structure is important for any project, even more so for a distributed project as the creation of clear roles, relationships and rules facilitate effective coordination and control (Casey & Richardson, 2006). The importance of clearly defining the roles and responsibilities within the organization is widely mentioned in the literature (Sudershana *et al.* , 2007; Berenbach, 2006; Bhat *et al.* , 2006). This allows everybody in the project to understand their and other people's roles and responsibilities, enabling the easy identification and finding of experts (Sudershana *et al.* , 2007).

As a general rule, teams should be constructed based on their known expertise, and not be split over more than two geographical areas (Lings *et al.* , 2007). However, in some cases it may be useful to set up mixed teams from different countries in order to integrate the different cultures (Ebert *et al.* , 2001). In some cases, the lower value-added work is usually done where it is cheapest (Lings *et al.* , 2007). This might, however, have an effect on off-site team members's motivation.

The size of the teams, the organization and the units of work should also be considered. Komi-Sirviö & Tihinen (2005) suggest that teamwork will be improved if the number of people involved in the project is under 20. The project should be divided into appropriate units of work, i.e. not too small or too large. The smaller a unit of work is, the lower the gain is in efficiency (Lings *et al.* , 2007). When determining the team sizes, the team members's other responsibilities within the organization should be also accounted for. It is highly recommended that team members should be allocated for one project only. In a long-term study, it was shown that distributed projects with highly scattered resources were half as productive that projects with a fully allocated staff (Ebert *et al.* , 2001).

Several other factors should be also considered regarding the distribution rationale. For some projects, separating the development of new functionalities from maintenance work can be useful (Ebert *et al.* , 2001). Another useful approach is to let testers and developers work together. This impacts team collaboration and helps anticipating defects early on. In addition, the separation of engineers may encourage them to create their own practices, which will impact the overall code quality. (Cristal *et al.* , 2008) This may also happen if there is an uneven distribution of developers across sites. Instead, there should be a minimal critical mass of developers at each site, so that misunderstandings can be solved locally with the help of each developer's knowledge about the various parts of the system (Cataldo & Nambiar, 2009).

### 3.2.4 Deciding how to divide the work

The assignment of tasks does often not get the required attention during project initiation, as deciding how to divide the work across the sites can be difficult (Lamersdorf *et al.* , 2008). The made decision has to take into account the available resources at each site, their level of expertise, and the available infrastructure (Herbsleb & Moitra, 2001). One common tactic is to minimize the dependencies between the sites by splitting the projects into smaller, more independent and more manageable units of work (Komi-Sirviö & Tihinen, 2005), effectively reducing the amount of collaboration needed between sites (Lamersdorf *et al.* , 2008). As another general advice, the work should not be divided before the larger context, team and environment are clearly understood (Gotel *et al.* , 2008).

Based on the existing literature, there are four common ways of dividing the work between sites:

- **Functional areas of expertise**: The work is divided so that expertise for a specific function is located at a single site (Grinter *et al.* , 1999). Co-location of all the experts of a specific domain helps with problem solving (Grinter *et al.* , 1999), and makes task assignment easier (Lamersdorf *et al.* , 2008). This approach has also the benefits of a larger pool of experts and better load balancing, making it easier to develop and enhance expertise as all the experts are co-located (Grinter *et al.* , 1999).

One disadvantage is, however, that adding new functionalities might require the use of experts from several different sites, effectively increasing the amount of needed collaboration between sites (Mockus & Weiss, 2001).

- **Modularization of product structure**: The product is divided into parts, with the various product parts developed at different sites. This results in an organization structure that mirrors the product structure (Herbsleb & Mockus, 2003). This approach requires a logical and modular architecture (Herbsleb & Grinter, 1999). The architecture must be well designed as it affects areas such as requirements, software configuration management, and interface management (Mettovaara *et al.* , 2006). One benefit of this approach is that the different modules can be developed using different tools, practices and processes (Grinter *et al.* , 1999). Disadvantages of this approach can include a difficulty of creating cross-site teamness and integrating the modules once they have been developed (Conchuir *et al.* , 2006).

- **Process life-cycle phases**: The work is handed off to another site after the completion of a specific process life-cycle phase. For example, the design and coding of a product can be done at a different site than the testing of the product. (Herbsleb & Mockus, 2003) As such, all the experts of a specific process life-cycle phase can be located to a single site (Mockus & Weiss, 2001). The obvious disadvantage is that there is a need for a great amount of collaboration between the handoff points (Grinter *et al.* , 1999).

- **Local customization**: An approach where the core product is developed at a central site, with other sites customizing the product for a local market (Herbsleb & Mockus, 2003). An example of such a customization is the translation of the documentation and user interface to a local language. The main benefit of this approach is a high awareness of the local customers's needs (Mockus & Weiss, 2001). In addition, it gives the customers of a local site an efficient way of submitting requests and bug fixes (Grinter *et al.* , 1999). One disadvantage of this approach is that there might be a need to maintain experts in all the domains at a local site (Mockus & Weiss, 2001).

### 3.2.5 Establishing trust between sites

Establishing trust between sites is a necessity for global software development. It is even more needed if a long-term relationship is to be established. (Hofner *et al.* , 2007) Establishing trust can, however, be quite difficult because of the geographical and cultural distances between the sites. Nevertheless, trust is required from both on-site and off-site. On-site needs to trust that off-site is able to deliver on time and up to quality standards, and that any problems will be communicated quickly. Off-site needs to trust that on-site is reasonable and helpful about problems if they are communicated honestly. (Herbsleb *et al.* , 2005)

Hofner *et al.* (2007) note that the main challenge is getting the buy-in from the on-site development team. On-site developers are afraid of losing their jobs (Herbsleb *et al.* , 2005), as the on-site developers are forced to train their cheaper off-site counterparts (Ebert *et al.* , 2001). As such, it is important to introduce an environment where it is not "us" or "them", but "we" (Hofner *et al.* , 2007). This can be done by emphasizing the role of everybody to create an atmosphere of partnership (Gotel *et al.* , 2008), and to involve members from both sites in the decision making processes (Damian, 2008).

## 3.3 Initiating a distributed project

### 3.3.1 Appointing an experienced leader

A globally distributed software project should have one identified project leader who is fully responsible for achieving project targets (Ebert *et al.* , 2001). This project leader should be supplemented with both team and local project managers, even if their responsibilities are overlapping (Lings *et al.* , 2007). Strong leadership that creates and embeds a proper culture into the project is required for distributed development to be successful. As such, a leader that has broad global and cultural perspectives should be chosen. (Hofner *et al.* , 2007)

Choosing a leader for the project should be done carefully and in the very beginning of the project. The chosen leader should have a broad experience with various offshore collaborations (Leszak *et al.* , 2007), have strong communication skills, have exposure to different cultures (Hofner *et al.* , 2007), be able to apply appropriate management styles towards both on-site and off-site coworkers (Lindqvist *et al.* , 2006), and have at least several years of industrial experience (Cusick & Prasad, 2006).

One of the most important tasks of a leader is to create a culture that encourages efficient communication between sites (Lindqvist *et al.* , 2006). Other tasks of a leader consist of providing strong support to the off-site teams so that corrective measures can be initiated early on (Hofner *et al.* , 2007), creating interdependencies and scheduling meetings between sites to encourage communication (Lindqvist *et al.* , 2006), establishing clear roadmaps and translating the business objectives into specific technical objectives (Narayanan *et al.* , 2006), and providing detailed milestones with clear metrics for tracking purposes (Herbsleb *et al.* , 2005). In some cases it may also be useful for the leader to be prepared to micromanage the off-site staff (Herbsleb *et al.* , 2005).

### 3.3.2 Recruiting new team members

Recruiting the best and most highly skilled people to the project is often seen as very important, as the availability of skilled competence is often seen as a reason why organizations engage in global software development. This allows an organization to focus on its core business and buy other competencies outside of the organization when needed (Mettovaara *et al.* , 2006). Mettovaara *et al.* (2006) further note that even though many organizations are looking for highly skilled experts, the low cost of personnel is usually the main driver.

All staffing decisions should be carefully reviewed and approved (Jensen *et al.* , 2007). The candidates should be carefully reviewed and only the well-qualified should be chosen (Cusick & Prasad, 2006). Jensen *et al.* (2007) suggest even to reserve the right to veto any staff assignment proposals made by the off-site organization. As it can be quite difficult to evaluate the quality and skills of developers at remote sites, the resumes of developers should be examined in order to avoid exaggerated claims (Herbsleb *et al.* , 2005). Khan *et al.* (2009) suggest to look for highly skilled experts with degrees in Computer Science, Engineering, Management or similar fields. The potential recruits should be also interviewed, especially the senior project members (Hofner *et al.* , 2007).

Being able to get appropriately skilled staff on short notice is becoming more difficult. As such, the project's resource needs must be forecast in advance (Cusick & Prasad, 2006). In addition, some time must be reserved for the training of the new recruits (Hofner *et al.* , 2007). One other common problem to look out for is that the stated qualifications and academic expertise do often not result in the expected productivity. A common pitfall when recruiting more off-site staff is to minimize the use of critical on-site resources. These resources are much needed for knowledge transfer, training and follow-up of the off-site team. Without these on-site resources, the off-site team is not able to deliver satisfactory quality. (Jensen *et al.* , 2007)

When the same set of people stay in the project for a longer period of time, working on the project becomes much easier (Mettovaara *et al.* , 2006). However, some GSD projects tend to suffer from a high turnover rate. As training of project members is quite expensive, there should be an attempt to keep the project members for a long-term period. This can however be in conflict with what most project members see beneficial for their own careers, as most of them might prefer to work a short duration on multiple projects in order to build a diverse experience. (Jensen *et al.* , 2007) The overall goal should be to the continuity of the off-site staff (Cusick & Prasad, 2006). This can be achieved by providing good career advancement opportunities (Jensen *et al.* , 2007), high-end work with advanced technologies (Kobitzsch *et al.* , 2001), and effective team integration with strong leadership (Hofner *et al.* , 2007).

### 3.3.3 Establishing efficient working practices

Before the actual work can begin, efficient working practices and software development processes must be established. It may be useful to provide an initial guideline to project members in order to foster a pro-active learning environment (Prikladnicki & Pilatti, 2008). This guideline should focus on topics such as coding standards (Cusick & Prasad, 2006), reporting mechanisms and performance metrics (Bhat *et al.* , 2006), risk management, scheduling practices, error correction, and version control (Kobitzsch *et al.* , 2001). A good practice is to also incorporate a workspace where this type of context-specific information can be saved (Mohan & Fernandez, 2010). This allows the recording of decisions, and makes the documentation available to everyone (Herbsleb & Grinter, 1999).

Different process models between sites can generate severe difficulties. As such, establishing a common software development process model that is shared between all sites, facilitates the coordination, communication and control in the project (Hofner *et al.* , 2007). Three strategies can be used to establish a common process between the sites: forcing standardization, blending different process components from different sites into a new process, and imposing high-level guidelines (Prikladnicki *et al.* , 2003). However, Battin *et al.* (2001) suggest on using existing processes so that teams can start producing results immediately using a process that they are already familiar with. If a site is forced to use a common process that they are not familiar with, the added learning curve must be taken into consideration when making a decision regarding the used process model. In addition to a common process model, the teams should share a common vocabulary in order to improve the communication between sites (Hofner *et al.* , 2007; Battin *et al.* , 2001).

### 3.3.4 Early and frequent traveling

Traveling between sites should take place in every distributed project. It can be used as a way to synchronize activities, to strengthen morale, to lower socio-cultural distance (Lings *et al.* , 2006), to build cultural awareness, and to integrate the team (Lings *et al.* , 2007). The following phases of a distributed project are recognized as ideal for traveling between sites: project kick-off, training and knowledge transfer sessions, and design and integration phases (Lings *et al.* , 2007). As such, the spending of the travel budget should not be postponed (Herbsleb & Grinter, 1999). Traveling between sites can be expensive and requires careful planning, with visa restrictions further complicating and delaying planned travels (Lings *et al.* , 2007).

Planning is also required regarding who gets to travel. Lings *et al.* (2007) suggest of letting both sites to travel across the sites in order to build cultural awareness. They further suggest of including both short management visits and longer visits for the developers. Management visits are useful for taking care of issues related to the budget, schedule, risk management, and the planning of the next release. Furthermore, management visits increase the transparency between sites as the managers are able to have a better overall visibility of the project. (Bass *et al.* , 2007)

The frequency and timing of traveling is also widely discussed in the literature. Several studies suggest that team members should meet each other's face-to-face as early as possible (Mettovaara *et al.* , 2006; Herbsleb *et al.* , 2005; Bass *et al.* , 2007). Early traveling helps bringing the off-site team up to speed (Sureshchandra & Shrinivasavadhani, 2008). However, traveling should not only happen in the very beginning of the project, many studies have also emphasized the need for frequent traveling (Sudershana *et al.* , 2007; Hofner *et al.* , 2007; Mettovaara *et al.* , 2006). Frequent visits allow the project to celebrate its every success and to uphold a good team spirit (Hofner *et al.* , 2007).

### 3.3.5 Holding a face-to-face kick-off meeting

Using face-to-face kick-off meetings as a way of building trust and personal relationships in the beginning of a distributed project is widely emphasized in several studies (Mettovaara *et al.* , 2006; Berenbach, 2006; Bhat *et al.* , 2006; Raffo & Setamanit, 2005; Pyysiäinen, 2003). Without the initial formation of trust it is unlikely that team members can effectively communicate across sites (Raffo & Setamanit, 2005). A common kick-off meeting in the beginning of the project helps creating initial familiarity between project members and therefore also makes them more eager to engage in communication once the project members are distributed. (Komi-Sirviö & Tihinen, 2005)

Jensen *et al.* (2007) suggest that kick-off meetings should be organized at both on-site and off-site. The kick-off meetings should consist of social activities in order to encourage integration on a social level. Activities such as scheduled informal chats, exchanging country-specific gifts, giving materials about the involved countries and cultures, and the announcement of local holidays should take place in the kick-off meeting. (Gotel *et al.* , 2008) In order to create a first good impression, it may be advisable to pickup the team members from the airport when they arrive, drop them off at the hotel, and show them around the local city. Small considerations like these help building a deep and long-lasting relationship between the sites. (Hofner *et al.* , 2007)

### 3.3.6 Training of new team members

Training of project team members has been widely mentioned as one of the most useful practices in the beginning of a distributed project (Pyysiäinen, 2003; Prikladnicki *et al.* , 2003; Dubé & Paré, 2001). Training provided at the beginning of a project is especially important as most of the project members do not have the necessary background information to work efficiently on the project (Pyysiäinen, 2003). Nevertheless, training should not only be given in the beginning of the project. Prikladnicki & Pilatti (2008) suggest that there should be a periodical training program offered every three months to a group of new recruits. Providing training regarding business and domain knowledge is also important (Herbsleb *et al.* , 2005). This is discussed in section 3.3.7.

The main benefits of training are increased know-how (Tervonen & Mustonen, 2009), and an increased trust because of established relationships between individuals across the sites (Prikladnicki *et al.* , 2003; Bass *et al.* , 2007). As such, the training should be organized as co-located, where members from all sites can meet each others (Bass *et al.* , 2007). In cases where co-locating all the team members is impractical, providing additional training material in electronic format might be useful (Komi-Sirviö & Tihinen, 2005). The electronic training material can be further reused in subsequent training sessions.

The provided training should consist of both practical and theoretical training. Practical hands-on training allows the team members to play and have fun with the system, simultaneously improving the team members's morale (Bondi & Ros, 2009). Theoretical training makes the team members more eager and committed to follow correct procedures and principles (Bondi & Ros, 2009). Another type of training mentioned in the literature includes coaching. According to Ebert *et al.* (2001), intensive coaching is the most efficient way of providing training when looking at the time needed to detect and correct defects, suggesting that five percent of the total project effort should be spent on coaching.

The provided training should not only be of technical nature. The training sessions should consist of topics such as communication, tools, technologies (Bhat *et al.* , 2006), culture, language (Lings *et al.* , 2007), leadership, context sharing, project management (Prikladnicki *et al.* , 2003) and the used development process (Lings *et al.* , 2006). Moreover, the training should consist of project-specific norms for meetings, deadlines and commitments (Paré & Dubé, 1999).

Providing training regarding tools and technologies is important when new technologies are being introduced in a project. It is especially useful for project members to know how they are going to apply particular technologies during the project (Bass *et al.* , 2007). In addition, the technical training sessions should cover topics such as standards (Bass *et al.* , 2007), software tooling, software build process and infrastructure usage (Kobitzsch *et al.* , 2001). Bass *et al.* (2007) note the importance of developing a common understanding of how to use and apply the different tools and technologies in the project. Bass *et al.* (2007) suggest also of using a simplified version of the target system when holding the technical training sessions. This allows the project members to use the real tool infrastructure of the project, giving them a chance to already start thinking about applying the taught approaches on real problems. As such, the project team members can already start discussing about particular design decisions during the training sessions.

Cultural training allows the different sites to be able to understand their counterparts (Hofner *et al.* , 2007). Cultural training should always be provided, disregard of how experienced the project team members are (Dubé & Paré, 2001). The cultural training should consist of both national and organizational culture (Dubé & Paré, 2001), with covering differences in behavior, communication styles, how individuals respect time (Hofner *et al.* , 2007), normal working hours, how decisions are made, how work will be reviewed and approved, and how to resolve conflicts (Dubé & Paré, 2001). In addition, differences regarding when to work independently and the expectations of being proactive should be discussed. It may be also useful to let team members to present their own assumptions and values of their own cultures and societies. The team members should be also asked to try to overcome the belief that one culture is better than another (Hofner *et al.* , 2007). Lings *et al.* (2007) suggest the use of a cultural mediator to avoid problems regarding cultural issues. The use of a cultural mediator is discussed in section 3.3.8.

Language training in a foreign language, such as English, is highly recommended (Dubé & Paré, 2001). Problems with language can be one of the most noticeable problems in meetings. As such, one major benefit of providing language training is the increased confidence in team members, making them more comfortable to engage in asynchronous forms of communication (Lings *et al.* , 2007).

### 3.3.7 Providing knowledge transfer

Knowledge transfer, also known as competence transfer, encourages the sharing of information between team members and supports the learning from experience (Prikladnicki *et al.* , 2003). The main benefits are that the off-site team members become more informed, trained and motivated. The lack of communication and appropriate knowledge transfer are often seen as the main reason why distributed projects either fail or are delayed (Sudershana *et al.* , 2007; Herbsleb *et al.* , 2005). Without the proper knowledge of the products, domain, processes and quality standards, it will be difficult, if not impossible, for the off-site team members to successfully deliver sufficient quality on time (Jensen *et al.* , 2007).

Cusick & Prasad (2006) suggest that knowledge transfer should be a planned activity with clear deliverables. Careful planning is needed regarding when and where, and from whom to whom to transfer knowledge. For instance, if the architecture design of a system is done at a different site from where the implementation takes place, careful planning is required in order to ensure that the design rationales are correctly understood by all sites. (Komi-Sirviö & Tihinen, 2005)

As knowledge transfer is often organized as face-to-face, one of the drawbacks of it is the high involved cost (Kobitzsch *et al.* , 2001). Another common problem is that organizations do not promote a culture of efficient information sharing and usage. Project members are often not aware or trained about the benefits of an information repository (Prikladnicki *et al.* , 2003). In addition, engaging on-site team members into providing knowledge transfer to their off-site counterparts can be difficult in an environment where on-site team members are worried about losing their jobs. As such, personal relationships between on-site and off-site team members are required before the knowledge transfer can begin. (Jensen *et al.* , 2007)

The following knowledge transfer techniques are mentioned in the literature:

- **Face-to-face** knowledge transfer consists of sessions where team members from all sites participate in the knowledge transfer. This approach is often regarded as the fastest way of solving a design problem or obtaining an answer to a specific question. One of the drawbacks is the involved high cost. (Komi-Sirviö & Tihinen, 2005)
- **Frequent rotation of team members** between on-site and off-site every three to six months allows the project team members to gain knowledge much faster because of the frequent interaction between on-site and off-site team members. (Sureshchandra & Shrinivasavadhani, 2008)
- **Design documents** used for knowledge transfer is often regarded as a slow and laborious process. Moreover, it is often difficult to achieve the correct level of detail and content. (Komi-Sirviö & Tihinen, 2005)

Knowledge transfer is not something that should only happen in the beginning of a distributed project. Kobitzsch *et al.* (2001) suggest that knowledge transfer should be a continuous activity. They further suggest on using an approach where key off-site team members are first trained at on-site, and then further training sessions are organized at off-site.

### 3.3.8 Designating liaisons

A liaison or a cultural mediator is a team member from one environment spending time in another, and thus becoming a communication link between the sites (Lings *et al.* , 2006). Designating liaisons introduces several benefits to the project such as an increased awareness of the other sites's culture (Lings *et al.* , 2007), improved personal relationships between team members (Bass *et al.* , 2007), faster problem solving (Kobitzsch *et al.* , 2001), an improved ability to see issues from the client's perspective (Heeks *et al.* , 2001), and a better integration of the distributed teams (Bass *et al.* , 2007). Moreover, the liaison can help to set correct expectations at both sites and help establishing a common vocabulary (Hofner *et al.* , 2007).

The duties of a liaison consist of facilitating communication, bridging cultures, mediating conflicts and resolving cultural miscommunications. As such, liaisons are usually expatriates or active travelers with broader global perspectives. (Carmel & Agarwal, 2001) In addition, liaisons can be useful for setting up direct contacts between on-site and off-site team members, as in most cases the liaisons know most of the people at both sites (Jensen *et al.* , 2007; Sudershana *et al.* , 2007).

One common problem in designating a liaison is that it may be hard to find a good candidate that is willing to relocate himself for a longer time period (Kobitzsch *et al.* , 2001). Bass *et al.* (2007) suggest that the typical length of a delegation is usually one to two years, whereas Lings *et al.* (2006) suggest that the liaison could be even relocated for the entire length of the project.

### 3.3.9 Selecting tools and setting up infrastructure

Before the actual work in a distributed project can start, it must be ensured that the appropriate tools, technologies and infrastructures are accessible and compatible across the sites (Dubé & Paré, 2001). The used tools and infrastructure should be common for all sites in order to make collaboration between sites easier. However, this might be costly if the other sites are already using different tools (Lings *et al.* , 2007). The learning curve of using a new tool can be decreased by using globally accepted tools (Sudershana *et al.* , 2007). Furthermore, mirrored infrastructure environments tend to require lots of specification and time to setup, which is the reason why setting up the infrastructure should be started early (Cusick & Prasad, 2006).

Sufficient tools and infrastructure should be set up for communication, video conferencing, shared workspaces and global software libraries (Ebert *et al.* , 2001). In addition, a shared repository should be set up where documents and artifacts can be maintained. This improves the visibility of each artifact's current status. (Lings *et al.* , 2007)

Careful planning is also needed regarding the network connections between the sites, as bandwidth capacities and cost structures for network access can be quite different in various countries (Dubé & Paré, 2001). As infrastructure is a necessity for any project, network and connection issues should be solved early on (Cusick & Prasad, 2006). If network connections are unreliable and get disrupted, the team members become frustrated and their efficiency drops radically (Dubé & Paré, 2001).

One common strategy is to also develop a project home page or set up a wiki that summarizes project information, progress metrics, planning information, and team-specific information (Ebert *et al.* , 2001). There should be information about national holidays and a maintained list of team members and their roles in the project (Lings *et al.* , 2007). In addition, Herbsleb *et al.* (2005) suggest having a photo gallery of project members to help people get a sense of those who they are collaborating with. They further suggest on printing out this photo gallery and put it up in every team room.

### 3.3.10 Continuing by growing gradually

Depending on the organizational strategy, the next steps should focus on the reduction of on-site overhead by gradually transferring more responsibilities to the off-site team after the distributed project has successfully been initiated (Jensen *et al.* , 2007). The major benefits of the GSD project will start to realize in the long-term view. After the relationship between on-site and off-site matures, the off-site team will start to deliver consistently on time, meeting the quality requirements (Narayanan *et al.* , 2006). After the working practices and processes have matured, more and more responsibilities should be transferred to off-site in a gradual fashion.

## 3.4 Summary of findings

Table 3.1 presents a summary of the findings from the literature study.

Table 3.1: Summary of findings from the literature

| PREPARING FOR A DISTRIBUTED PROJECT | |
|---|---|
| Developing and communicating a GSD strategy | • Agree and communicate about the project's goals, targets and objectives to ensure that all commitments are clearly understood by all involved sites.<br>• Take a long-term strategic view, as most benefits will be realized in the long-term period.<br>• Communicate about on-site team members's career paths as they may have a fear of losing their jobs. |
| Early planning and preparation | • Engage the team members in the project by building a project vision collaboratively.<br>• Have a concrete plan on what to distribute and why, followed by a detailed roadmap, which defines how the project will grow in terms of team size and responsibilities. |

| | |
|---|---|
| Structuring the teams and the organization | • Create clear roles, responsibilities and rules to facilitate effective coordination and control.<br>• Clear roles allows the easy identification and finding of experts within the organization.<br>• Do not allocate team members to several projects within the organization.<br>• Consider separating the development of new functionalities from maintenance work.<br>• Allowing testers and developers to work together to increases collaboration and helps anticipating defects early on. |
| Deciding how to divide the work | • Minimize dependencies between sites by splitting projects into smaller, more independent and more manageable units of work in order to reduce the amount of needed collaboration.<br>• Do not divide the work before understanding the larger context, team and environment where the project takes place. |
| Establishing trust between sites | • Introduce an environment where it is not “us” or “them”, but “we”.<br>• Emphasize the role of everybody and create an atmosphere of partnership by involving team members from both sites in the decision making processes. |

| INITIATING A DISTRIBUTED PROJECT | |
|---|---|
| Appointing an experienced leader | • Have one identified project leader who is fully responsible for achieving project targets.<br>• Choose a leader with broad global and cultural perspectives.<br>• Choose a leader that has experience in various offshore collaborations, strong communication skills, and exposure to different cultures. |
| Recruiting new team members | • Review, and possibly interview, all eligible candidates, with only choosing the well-qualified.<br>• Forecast resource needs early, as it may be difficult to find appropriately skilled staff on a short notice.<br>• Avoid minimizing critical on-site resource when recruiting for more off-site team members.<br>• To avoid a high turnover rate, provide off-site team members good career advancement opportunities and high-end work with advanced technologies. |
| Establishing efficient working practices | • Consider providing an initial guideline to project members in order to foster a pro-active learning environment.<br>• Establish a common software development process model that is shared between all sites to facilitate the coordination, communication and control in the project. |

| | |
|---|---|
| Early and frequent traveling | • Early traveling can be used as a way to synchronize activities, to strengthen morale, to lower socio-cultural distance, to build cultural awareness, and to integrate the team.<br>• Allow team members from both sites to travel in order to build cultural awareness across sites.<br>• Invest in both short management visits and longer visits for technical staff and developers. |
| Holding a face-to-face kick-off meeting | • Hold a face-to-face kick-off meetings as a way of building trust and personal relationships in the beginning of a distributed project.<br>• Include social activities in order to encourage integration on a social level. |
| Training of new team members | • Provide training to new team members so that they can have the necessary background information to work efficiently on the project.<br>• Include both theoretical and practical training sessions.<br>• Provide training regarding topics such as communication, tools, technologies, culture, language, leadership, context sharing, project management and the used development process. |
| Providing knowledge transfer | • Provide off-site team members proper knowledge of the products, domain, processes and quality standards.<br>• Organizing face-to-face knowledge transfer sessions can be expensive, and thus needs careful planning regarding when and where, and from whom to whom to transfer knowledge. |

| | |
|---|---|
| Designating liaisons | • Designate a liaison to help with facilitating communication, bridging cultures, mediating conflicts and resolving cultural miscommunications.<br>• Assign persons as liaisons who are expatriates or active travelers with broader global perspectives.<br>• Liaisons can be useful for setting up direct contacts between sites, as in most cases the liaisons know most of the people at both sites. |
| Selecting tools and setting up infrastructure | • Ensure that the appropriate tools, technologies and infrastructures are accessible and compatible across the sites.<br>• Use common and shared tools and infrastructure to make the collaboration between sites easier.<br>• Develop a project home page or set up a wiki where one can summarizes project information, progress metrics, planning information, and team-specific information. |
| Continuing by growing gradually | • Consider the reduction of on-site overhead by gradually transferring more responsibilities to the off-site team. |

# Chapter 4

# Results

This chapter presents the findings from the studied case projects. The chapter begins with an introduction of the case organization and the studied case projects in section 4.1. Section 4.2 presents findings related to the preparation of a distributed software project. After this, the chapter continues with section 4.3 presenting solutions and useful practices for initiating a distributed software project. Finally, section 4.4 presents a summary of all the findings from the studied case projects.

## 4.1 Case projects

All the studied case projects in this study were selected from one single case organization. This case organization is a global player in the software industry with offices in several countries. The case organization consisted of several sub-organizations. The case organization's strategy was to gain growth by setting up offshoring centers in countries where talented and cheap labor existed. As such, some of the findings may only be applicable to projects operating in a similar organizational environment.

The studied case projects consisted of two successful and two challenged cases. Case Alpha and Case Delta were considered as successful, whereas Case Beta and Case Gamma were considered as challenged. Three of the studied case projects had their off-site team located at a software center in Czech Republic, whereas one of the case projects had its off-site team located in Malaysia. Other differences between the studied case projects can be seen from table 4.1. One notable difference between the case projects was how the teams were structured. Both Case Alpha and Case Beta had team structures based on the project's life cycles, i.e. they had teams such as testing and development. Case Gamma and Case Delta differed in the sense that the teams were divided based on the product's structure. Thus, their teams consisted of both testers and developers.

Table 4.1: A comparison between the studied case projects

| | Alpha | Beta | Gamma | Delta |
|---|---|---|---|---|
| On-site location | Finland | Finland | Finland | Norway |
| Off-site location | Czech Republic | Czech Republic | Czech Republic | Malaysia |
| Time-zone difference | 1 hour | 1 hour | 1 hour | 7 hours |
| On-site team size | 6 | 20 | 17 | 20 |
| Off-site team size | 26 | 11 | 19 | 20 |
| On-site/Off-site activity | 20% / 80% | 60% / 40% | 50% / 50% | 50% / 50% |
| Team structure | Life-cycle teams | Life-cycle teams | Functional teams | Functional teams |
| Cross-site teams | No | No | Yes | Yes |
| Process model | RUP | RUP | RUP | Scrum |
| Iterative development | Yes | Yes | Yes | Yes |
| Iteration length | 2-4 weeks | 4 weeks | 3 weeks | 4 weeks |
| Project kick-off | 2004 Q4 | 2006 Q2 | 2003 | 2003 |

### 4.1.1 Case Alpha

In Case Alpha, two different products were developed for a global client in the telecommunications industry. One of the major benefits in this project was that there was an understanding client with previous experiences from GSD. For example, the client understood that the ramp-up phase of a distributed project is an important phase that requires a lot of investment and time.

Case Alpha was launched in the late 2004, at the same time as the first few people at off-site were recruited into the project. These few developers started their work by reading the existing documentation and learning how the system works. After a couple months, the project was officially launched and more people were recruited to the project. All the growth in project personnel was coming from the off-site location, a strategy that was seen as a very good one.

RUP was taken into use as a high-level process model in the middle of 2005.

Risk-driven development, risk management, iterative development, nightly builds and daily meetings were a few of the introduced changes. Implementing all the changes took around one year. During the introduction of these changes, the off-site team was working on a very limited version of the product, an approach which allowed them to simultaneously focus on implementing efficient working practices.

After the off-site team had proven that they were capable of successfully working in the project, more responsibilities were given to them in a gradual fashion. For example, after enough trust had been formed between the sites, the off-site team was allowed to be in direct contact with the customer. This had a serious boost in the off-site team's morale.

In addition, the project had an experienced leader at on-site, as described in section 3.3.1. He realized the importance of forming trust and understanding between team members by introducing active traveling between sites, socializing events such as bowling and going for a dinner, and team bonuses. In addition, he invested heavily in a good kick-off by providing training and knowledge transfer to team members. He also managed to solve some of the problems encountered in the beginning of the project. For example, the off-site team members had problems in taking new responsibilities, making commitments and reporting about problems.

### 4.1.2 Case Beta

In Case Beta, a system was built to a customer that operates in the financial market in Europe. As the system was connected to many other related banking systems, the system was seen as very critical. This case differed from the other cases in the sense that the developed system was completely based on an old version of the system. As such, a legacy system was coded again with the use of newer technologies. The old version of the system was developed by the organization's on-site developers. These on-site developers were, however, quite busy with other projects in the organization, and were thus not able to give enough support for their off-site team members.

The project started with a ramp-up period in early 2006. A kick-off session together with a two-week training period for the off-site team was held in Finland. Training regarding the used development framework was provided. The off-site team members were introduced how the system can be tested, debugged and coded with the use of the development framework. However, the length of the training period was considered too short, as team members were not given enough time to socialize and establish personal relationships with one another.

Case Beta used a single-site team structure. The model used for dividing the work was process life-cycle phases, as described in section 3.2.4. The handovers from different process phases were done in a waterfall-like way. This had a serious disadvantage of causing some of the teams to become bottlenecks during different times. However, the handover of work was managed through several face-to-face workshops, which was seen as a very useful practice. Later during 2006, iterative development was taken into use.

The project continued to suffer of several challenges. From the beginning of the project, there was a very confusing team and organization structure in use. This was partly because the on-site team consisted of several team members that were working in various projects in the organization. This resulted in that on-site was too busy in providing the required support and follow-up to off-site team members. Another major issue was the lack of a task management tool, which was something that the off-site team had been asking for over one year. However, this request had been continuously denied by on-site because of potential security issues. Moreover, most of the customer representatives were not proficient in English and all the existing documentation was in Finnish. Together with all these issues, the project was considered to be severely challenged.

### 4.1.3 Case Gamma

In Case Gamma, a system for enterprise resource and customer management was developed. This system was sold to and used by a small number of Finnish customers, with all of them needing customer-specific tailoring. In addition to customer-specific tailoring, a majority of the work consisted of new feature development. The project was distributed mainly in Finland and Czech Republic, with some management also located in Sweden.

The project started with a kick-off in late 2003. An off-site team of two developers got involved in the project in 2004. After this, the size of the off-site team was gradually increased until 2005. In late 2005, the project got a new customer which required additional new resources. However, this customer eventually decided to discontinue as a customer in early 2007. This was one of the reasons why the project got severely delayed. During 2007, the project began to introduce some organizational and process changes. The main changes included the introduction of cross-site teams, weekly meetings and iterative development.

The project suffered from many difficulties. One of the bigger issues was the involved languages. The customers wanted to communicate and have documents in Finnish, whereas the developers in Czech Republic needed specification documents in English. The business domain was also very complicated and hard for off-site developers to understand. Combined with the fact that both on-site and off-site had difficulties in communicating in English, there were several misunderstandings and a lack of communication throughout the project.

Another severe problem was the lack of structure. The organization structure at on-site was very unclear. Before the organization changes were introduced in 2007, it was even unclear who was leading the project.

Another issue was that the goals and targets of the project were not collaboratively decided. There were clear disagreements whether to build a customer-specific system or a productized product. While the customer analysts at on-site tried to be more customer-oriented, the software developers at both on-site and off-site were trying to build a more standard product. This resulted in several disagreements, and thus the only focus was to concentrate on new feature development, with no time given to stabilize the product.

### 4.1.4 Case Delta

Case Delta differed from the other case projects in many ways. The involved sites were different from the other case projects, with on-site being located in Norway and off-site located in Malaysia. In addition, there were four other sites in various locations around the world that were involved in maintenance work and customer-specific customization. The project was developing an enterprise resource management system for a specific industry.

Before the project was initiated, there had been many years of customer-specific development and consulting. The project got started in 2003 with an existing version of an old product. This product was baselined and scaled down to minimum. After the completion of the core baseline, the off-site team was involved in the project, with the development responsibilities being transferred from on-site to off-site.

A waterfall-like development model was used from the beginning of the project. Eventually, this development model was changed to Scrum, which is an iterative development model for agile software development. This allowed a more rapid response time to customer change requests. The main change was the introduction of iterative development and the removal of project and test manager roles. One of the major benefits of Scrum was the heavily increased communication because of the daily Scrums.

## 4.2 Preparing for a distributed project

This section presents practices and ideas that should be considered before initiating a distributed project. These practices include different approaches on how to divide the work and responsibilities between sites. As in any software project, careful planning should take place in order for the distributed project to have a chance of success, even more so in a distributed project.

### 4.2.1 Evaluating the benefits of offshoring

Potential benefits of offshoring must be carefully evaluated before going global. These potential benefits should be weighed against the potential costs of distributing a project to several sites.

The most common reason for the studied case projects to offshore part of their activities to other countries was the associated lower cost. However, as the set up costs of a distributed project tended to be relatively high, a lot of time and effort had to be spent on several activities in the beginning of the case projects. Thus, according to some interviewees, a long-term view has to be taken in order for any cost savings to be realized. Because of this, the length of the project should be estimated. Too small projects may not be suitable for offshoring.

The amount of work and responsibilities that can be transferred from the on-site location to the off-site location should also be evaluated. Being able to move as much of the work as possible from the on-site location to the off-site location will naturally introduce more cost savings. In the studied case projects, common problems why work could not have been moved to the off-site location were:

- **Lack of language skills at off-site.** In Case Beta and Case Gamma, many of the responsibilities, such as requirements management, could not be moved from on-site to off-site as there was a language barrier between the sites. Most of the existing documentation was in a language only understood by the on-site team. Moreover, the customers of these two projects preferred to also speak in this language. As such, the existing artifacts should be in a language that is understood by both sites.

- **Lack of documentation.** In Case Beta, the system to be built was based on other existing systems. These systems were poorly documented and in most of the cases the existing documents were not in English. This, combined with the fact that most knowledge of the existing systems was tacit knowledge possessed by the on-site team, makes knowledge transfer from on-site to off-site even harder.

In Case Alpha, an impressive amount of work and responsibilities was transferred from on-site to off-site. Around 80% of all the project activities took place at the off-site location, whereas in Case Beta, for example, this number was only 40%. Offshoring was a particularly good strategy for Case Alpha as it was possible to transfer most of the activities from on-site to off-site. This was possible because of several conditions:

- **Existing standards as requirements specifications.** Case Alpha differed from all the other studied cases in the sense that most of the requirements for the product were coming from existing network communication protocol standards. This allowed the off-site team to gain the same domain knowledge of the product as the on-site team.
- **Small on-site team.** The size of the on-site team was very small to begin with in Case Alpha. This allowed the growth of the project personnel at the off-site location, and to keep all the personnel, expertise and knowledge at the on-site location. The size of the on-site team was six throughout most of the project.
- **A common language between all sites.** When most of the communication between all the involved sites is done with the same language, there are fewer difficulties in transferring responsibilities from one site to another. The main language used in Case Alpha was English. One of the recruitment criteria for the off-site team members was that they were able to speak good English. Moreover, the on-site team consisted of native English speaking team members as well. This forced the on-site team to also do most of their intern communication in English. Lastly, as all the specifications were in English, it was possible to transfer most of the requirements management and design work from on-site to off-site.

Whereas Case Alpha managed to locate most of the activities to off-site, many of the other studied case projects did not. In Case Beta, for example, the off-site team consisted of only 9 developers and 1 project manager, whereas the on-site team had between 14 to 20 full-time team members throughout the project. Indeed, more of the work and responsibilities should have been transferred from on-site to off-site in order to make the offshoring more profitable. Involving more sites in a project adds usually more overhead related to communication and collaboration. Therefore, enough work should be transferred from on-site to off-site to compensate for this overhead. Otherwise, it may be advisable to keep the project co-located.

Good language and communication skills are essential when working in a global software development project. Not only should the off-site team members's language skills be evaluated, but also the on-site team members's. In two of the studied case projects, namely Case Beta and Case Gamma, the on-site team members had difficulties in expressing themselves in English. Moreover, most of the customer contact persons were not capable of speaking English. As such, careful evaluation is needed in order to find out if the on-site team, the off-site team and the customer representatives are able to work in a global environment.

To summarize, careful evaluation of all teams's communication, language and working capabilities in a global environment is required.

### 4.2.2 Using cross-site teams over single-site teams

Most software development works tends to be of very knowledge-intensive. In situations where the off-site team is heavily dependent on the knowledge of their on-site colleagues, an organization structure with cross-site teams should be considered, as was done in Case Gamma and Case Delta. Cross-site teams are teams that consist of both on-site and off-site team members, whereas single-site teams consist of members from only one single site. On the contrary, single-site teams are suitable for projects where both sites have already sufficient knowledge of the product and its domain, making the required communication and collaboration between the sites low.

Using an organization structure with cross-site teams has some requirements for the product's structure as well. Case Gamma had introduced the cross-site teams under the term ”module teams”, where each team was responsible for one module in the product. In order to be able to efficiently use an organization structure of this type, it must be ensured that the product can be divided to several independent modules, so that each team can work on their artifacts without having too many dependencies to other teams. However, some dependencies in the product's structure did exist in Case Gamma. Because of this, some of the teams had to collaborate together at times. In practice, this was solved by allowing some team members to be part of several module teams.

Cross-site teams were taken into use during the middle of two of the studied case projects, namely Case Gamma and Case Delta. In both of the studied cases, the change towards cross-site teams was seen as positive. This change forced the team members from different sites to communicate with each other's more as they became more dependent on one another. In addition, the team members felt that they belonged to one single project team and that they could better trust their coworkers. The introduction of cross-site teams allowed the teams to have common goals and a shared vision, which made the team members more committed to the project, as noted by an on-site project manager:

> *"The project has been improving enormously, partly because of the improved communication. Earlier there was a lack of trust, as we had clearly separated on-site and off-site, with no common goals and targets. But now we have emphasized and tried to communicate the importance of one single project team with a common goal. And our team members have also noted a positive change. They now consider themselves to be part of the project." - On-site project manager, Case Gamma*

Traditionally, a team consists usually of a team leader who is responsible for the decisions and actions of the team, and who often also defines the goals, methods and working practices of the team. In Case Gamma, the team leader differed from the other team members only in the sense that he had the responsibility of progress reporting. However, one of the problems behind this type of thinking was that the off-site people had easily a hierarchical way of thinking, i.e. the off-site team leader was still in a too central position. For example, most of the communication that still took place was channeled through the team leaders. Nevertheless, according to one interviewee, this type of division into module teams had the expected outcomes:

> *"The responsibility for controlling has now spread to those module leaders. They know their areas and they know their tasks. They now have a close connection between off-site and on-site. They have now started to have weekly meetings and daily meetings when needed." - Off-site project manager, Case Gamma*

Case Delta had taken the idea of module teams a bit further, giving each team a complete decision authority. These teams were given a self-responsibility to make them more independent from other teams. As each team had a Scrum-like product owner that was able to decide about all the changes to the specific module, any questions regarding the tasks or business domain of this module could be directed directly to the product owner. In most cases, the product owner was able to directly answer the team's questions without delay. In addition to these decisions, the whole team was given freedom to decide on the team's working practices. For example, it was up to the module team to decide how to implement quality assurance practices. This approach was seen as very successful as it enabled fast decision-making and avoided the escalation of decisions to top management. It further increased the trust between sites as there was an attempt to keep some level of decision authority at both sites, as each team was using a cross-site team structure.

In essence, there are several benefits of using a cross-site team structure, such as forced and frequent communication between team members, increased collaboration between the sites, and increased trust between the team members.

### 4.2.3 Starting with system testing

Based on the findings from Case Alpha and Case Delta, the testing of the product should be the off-site development team's first responsibility when offshoring an existing product for offshore development. By first testing the product, the team will be able to get more familiar with it and gain some vital business and domain knowledge of the product. It is logical to first give off-site the knowhow of how the system works and what it is supposed to do, rather than how to code the system. Thus, the off-site team will be able to know how the system should work from a user's point of view.

This approach was used in two of the studied case projects, namely Case Alpha and Case Delta. In Case Alpha, the first responsibility of the off-site team was to review and execute all the test cases in order to know how the system was supposed to work. This will obviously require that there exists a product and a set of test cases that can be executed. In case no test cases have been designed, it may still be reasonable for the off-site team to begin with the testing of the product. In this case, the first task of the off-site team would be to design test cases for the product. From a developer's perspective it is a very logical approach. It is unreasonable to expect a developer to know how to write code for a product that he does not know how to use, as noted by an off-site project manager.

> *"I think that there was a quite good idea about not starting with the pure development [of the system]. So basically, not saying "okay, now we have it, so let's do some development", but rather looking on it from the user's point of view. It means starting with testing, which means understanding what the system is supposed to do, and not how, from the start." - Off-site project manager, Case Alpha*

Testing the system first before starting with development gives some well-needed skills and experience regarding the product for the off-site team members.

### 4.2.4 Locating testing and development at the same site

When planning the division of work and responsibilities between sites, special attention should be put into the relation between testing and development. If possible, the testing and the development of the system should be located at the same site, as was done in Case Alpha and Case Delta. Because of this, Case Alpha and Case Delta realized several benefits, such as:

- enhanced communication between testers and developers,

- a decreased effort in transferring a testable product from development to testing,
- a potential increase in product quality, and
- it allows the testing of new features to begin earlier.

Indeed, some level of collaboration is usually needed between testers and developers. In two of the studied case projects, namely Case Beta and Case Gamma, there was a separate testing team at on-site while the development was located at off-site. As such, all testing was performed at on-site. On-site had a dedicated testing team with the responsibility of designing and executing test cases. This generated several problems regarding the collaboration between the sites. As a first, the testing teams at on-site reported that most of their time was spent on documenting and reporting very trivial bugs. Instead, these trivial bugs could have been easily revealed at the off-site location had there been someone that was able to perform testing. As such, it would have been ideal to do at least some amount of testing at off-site to get rid of the most trivial bugs, as suggested by an on-site test manager:

> *"I think they [off-site] should have their own dedicated team of testers, because the most important thing is to minimize the amount of trivial bugs before the system is sent to us [on-site] for testing... The amount of workload needed to actually fix the bug is small, but the amount of extra work and communication needed to observe, track, process and document these defects is too much. The biggest problem is that all our effort goes into taking care of the trivial defects." - On-site test manager, Case Beta*

Secondly, the off-site development teams were often unable to ship a testable piece of work on schedule, which caused the off-site development teams to become bottlenecks. From the findings of these two case projects, when testing and development teams are located at different sites, the testing of the product will usually not begin before the development teams have completed their work and handed it over to the testing teams. Therefore, locating the testing and development teams at the same site makes it easier to begin testing even while the development of the product still continues. In addition, the testing of new features may begin earlier. This becomes easier if the people doing development are co-located with the testers, as the communication will be much richer when testers and developers are able to engage in face-to-face communication. For example, the developers are capable of directly pointing out which features can and cannot be tested, and similarly, the testers can describe the found defects in a step-by-step fashion to the developers. This difficulty was realized in Case Beta, as described by an on-site test manager:

> *"And then regarding the testing, one of our main challenges has been to describe the defects into Test Directory in a step-by-step fashion so that an off-site tester is able to reproduce the bug." - On-site test manager, Case Beta*

### 4.2.5 Separating maintenance from development work

In Case Alpha and Case Delta, there was a need to offer maintenance development, such as customer-specific configuration or development, or development done in the customer's native language. For these projects, having a separate maintenance team apart from the development teams introduced several benefits, such as improved response time to change requests and a decreased amount of needed collaboration between sites. As support requests required a lot of collaboration with external parties, such as customers, locating the support and maintenance teams at on-site was considered as a good approach.

A separate maintenance team can quickly react to support requests and thus be able to give a more rapid response time to change requests. As a separate maintenance team is often not dependent on the activities performed by other development teams, the maintenance team can solely focus on their tasks, allowing the other development teams to solely focus on developing the main branch.

In Case Delta, a separate maintenance team was located at on-site. This maintenance team had a different development cycle than the other teams. Their development cycle was a short two-week cycle, which allowed them to ship a maintenance release every two weeks. The off-site location consisted of only development teams that were building the main branch of the product. One of the reported benefits of this approach was that the amount of needed collaboration between the sites decreased as the frequent change requests did no longer require any work from the off-site team. Furthermore, the other development teams were given a peace-of-mind, since they could completely focus on new feature development.

### 4.2.6 Structuring the teams and the organization

Based on the findings from the studied case projects, a clear organization structure allows the project members to better understand everybody's responsibilities. In order to efficiently plan the division of work and responsibilities, there should be a clear organization structure that is clearly communicated to everyone involved in the project. A clear organization structure increases communication as it allows project members to know whom to contact.

When communicating the organization structure to project members, a clear division between the sites should be avoided. Instead, a feeling of a single project team in order to increase trust between sites should be communicated. In most of the studied case projects, a project website was used to communicate the organization structure together with all the project members's contact information.

Case Beta had a complete lack of hierarchy and structure. The project was organized simply into two separate teams, namely on-site and off-site. In this type of team structure, most of the communication between the sites was channeled through the project managers of each site. Because of this, there was a lack of direct communication between on-site and off-site developers. Moreover, the off-site developers were unsure of whom had done the work at the on-site location, e.g. who had designed the analysis models and written the specifications from the on-site team. This made the off-site developers resistant from asking questions from on-site, as each question was always channeled through the project managers.

### 4.2.7 Motivating off-site with meaningful work

Starting a new distributed development project will always introduce new sites and people to the project. Gaining the trust of these people will naturally take some time. However, it is important to be open-minded from the beginning of the project and to be willing to listen to the ideas of the new recruits. In two of the studied case projects, there was a slight resistance in transferring responsibilities from on-site to off-site. It is important to give out some meaningful work and responsibilities to off-site in the early-life of the project in order to show trust.

In both Case Gamma and Case Beta, we encountered a severe resistance from the on-site team regarding transferring of main development responsibilities from on-site to off-site. In both of the cases, off-site had requested several times that the development responsibilities should be moved from on-site to off-site. The development responsibilities were eventually moved from on-site to off-site in one of the cases, as noted by an off-site team leader:

> *"It was funny, we were fighting to move the responsibility of development to off-site for like one year." - Off-site team leader, Case Gamma*

However, a one year delay was quite unnecessary. On-site's original plan was to use the off-site team simple as a labor force, with most knowledge-intensive work taking place at on-site. This type of labor work becomes eventually very demotivating for the off-site team members. As such, the relationship between on-site and off-site seemed more of a contractor-subcontractor relationship, where all decision-making took place at the on-site location. This indeed does have a demotivating effect on the off-site team members, as commented by one off-site team member:

> *"We have here at off-site thousands of ideas of how to improve the project. Every idea was declined. "We don't have time to do it." It is not as motivating as it should be ... They just said "now there is no time" – which means that there is never time. When you hear*

> *it again and again it is not motivating." - Off-site team member, Case Gamma*

To illustrate a more positive example, Case Alpha had understood the importance of being open-minded and trustful regarding their off-site counterpart. On-site had given off-site the responsibility of all software development activities, allowing the off-site team members to become experts in their field, and the on-site team members to solely focus on other activities. Below is a comment from an off-site team leader when asked about the lessons he had learned during the project:

> *"In my mind we should be here [at off-site] the experts for software development ... it should be up to us, how to manage the software development project. Concerning what on-site should be experts on, they should be the experts on what the customer wants." - Off-site team leader, Case Alpha*

The secret to Case Alpha's success was that most responsibilities were transferred from on-site to off-site in a gradual fashion, an approach which is discussed in section 4.3.7.

## 4.3 Initiating a distributed project

The previous section presented a number of issues that should be considered before initiating a distributed project. This section continues to present practices that should take place directly after the initiation of a distributed project.

### 4.3.1 Recruiting new team members

Starting a distributed development project will always introduce new sites and project members to the project. As with any software project, careful selection of the new team members should take place.

As a first, the on-site team should request the off-site candidates to write résumés for on-site's reviewal. According to the interviewees from Case Alpha, good criteria for selecting new recruits are previous experience with global software projects and the used technologies.

After the first round of selection has been completed, an interview with the selected candidates should be held. It is important to ensure that the interviewee is able to fluently speak a language that is common for both sites.

In Case Alpha, another interesting approach was also used for selecting some of the candidates. In this case, the project manager had asked recommendations from other qualified candidates. For example, if one candidate had good technical skills regarding a specific technology, he was asked to point out other developers he might know that occupy the same skills. Moreover, it might be viable to attempt to recruit developers that have previously worked together. This was done in Case Alpha by asking the potential recruits about people who they have worked with in the past.

### 4.3.2 Holding a face-to-face kick-off meeting

After the first set of off-site candidates has been recruited for the project, the project should start with a kick-off meeting. This meeting should be held face-to-face together with the whole project team. A face-to-face kick-off meeting will give the project team a chance to meet each others before starting to work together. When communication in a distributed environment, most people will usually feel awkward if they have not seen each others before in face-to-face. Thus, a kick-off meeting should include team building activities so that the project members can form basic social ties between one another.

A kick-off meeting is also a good opportunity to present some background information of the project to the whole project team. This includes information such as the project's goals, targets, customers, used technology, and the product. It is especially advisable to present this information for discussion to the whole project team, so that everybody gets a common understanding of what is to be done in the project.

In Case Gamma, there was no joint kick-off meeting between the on-site and off-site teams. In this case project, only the top management from each site had a meeting in the very beginning of the project. No off-site developers were invited to participate in this kick-off meeting. As such, several of the off-site developers complained that they were never given a proper introduction to the project, e.g. what the project is about, what the project's targets are, and what will be the next steps. Instead, they were told to read some practical guidelines of how to work on the project. This lack of introduction to the project and formation of social ties prohibited the sites from communicating efficiently with each others. In essence, there was a lack of communication between the on-site analysts and the off-site developers, as most communication was channeled through the project managers.

In most of the studied case projects, the length of the project kick-off meeting was no longer than a few days. Having only a few day project kick-off meeting with the whole project team might not be cost-efficient because of the related traveling costs. Therefore, it is advisable to integrate the project kick-off meeting with knowledge transfer sessions, which will be discussed in the next subsection.

### 4.3.3 Training of new team members

A newly involved site will often require some level of knowledge transfer before it is capable of working efficiently. Transferring knowledge from the on-site team to the off-site team is thus one of the most important activities that should take place in the very beginning of the project. These knowledge transfer sessions should include training regarding the business domain, the product domain, the used software process, the used development tools and frameworks. If there are huge cultural differences between the sites, training regarding these differences might be also worthwhile.

In Case Alpha, a 4-week long knowledge transfer session was held at the on-site location. These training sessions were both theoretical and practical, which allowed the off-site team to first get the big picture of how the system works, and then go into more details such as how code the system. The combination of both theory and practice was seen as good. The theoretical lectures allowed the off-site team to understand the main concepts behind the used technical frameworks, whereas the more practical hands-on approach allowed them to use these technical frameworks.

As the knowledge transfer period lasted for a total of four weeks, the project members had enough time to socialize and build personal relationships to one another. Social activities such as having a dinner and going to sauna were also included. Some of the project members saw the team building activities as more important than the training sessions, because these established relationships enabled a more efficient collaboration between the two sites once they got distributed. Below is a comment from an off-site tester regarding these trainings:

> *"It was really good that we had been three weeks at on-site, the whole testing team. We started with team building, which was really crucial for the success of this testing. Because after the sessions we were sitting in the sauna and discussing what we have learned. That was like really amazing, with ten hours per day of work, discussing about it and still learning. Even during these sauna sessions we learned a lot of things. And the second thing was that we actually learned something about this project. I think that this [training regarding testing] was less important than this team building activity." - Off-site tester, Case Alpha*

In this same case project, a few of the off-site team members would have preferred that there had been more preparations for the training sessions. It would have been useful to have training material that could have been distributed to the whole off-site team. Instead, most of the material that off-site gained from these training sessions were their own notes and presentation slides. Any prepared training material will also be useful for training new recruits in the later stages of the project.

Knowledge transfer sessions should not only be held in the beginning of the project, but also on an on-demand basis. In Case Beta, a handover workshop was held between the on-site design team and the off-site development team for each system that was sent for development to the off-site location. These handover workshops allowed the off-site development team to review any documents produced by the design team and to gain further knowledge about the system. Furthermore, as these handover workshops always included the relevant analysts and developers from both sites, the developers from off-site knew whom to contact once they got some additional questions.

As a contrasting example, in Case Gamma, no knowledge transfer sessions were organized in the beginning of the project. Instead, the off-site team received specification documents that they could study by themselves. Project members from off-site complained several times that they do not have enough knowledge of the business and product in order to complete their tasks. This was one of the most severe problems in the project. On-site management did not see it cost-efficient enough to organize knowledge transfer sessions for the off-site team. Below is a comment from an off-site team leader that emphasizes the need of knowledge transfers:

> *"Only sending technical documentation is not enough. It just could not work. It is not the way proper off-shore development should be done." - Off-site team leader, Case Gamma*

> *"We are only workers at off-site because we don't have the knowledge, and we have been complaining about that from beginning of the project that we would like to have knowledge transfer more often and so on. But this is not happening." - Off-site team leader,*

*Case Gamma*

One of the obvious drawbacks of organizing knowledge transfer sessions is the involved cost. It is not cheap to relocate one of the teams for a long time period, and to have part of the on-site resources to hold the knowledge transfer sessions. Therefore, careful planning of the sessions is required.

### 4.3.4 Designating liaisons

A liaison is a person that moves from one of the sites to another for a long-term period. Doing this might introduce several benefits, such as improved communication and higher transparency between sites. As the liaison will know most of the team members from both sites, he is able to work as a communication proxy between the sites. His long-term presence will moreover help the other site to become custom with the working habits and culture of the remote site.

In Case Beta, an off-site team members was moved to the on-site team. This person had several contacts at the off-site location and knew most of the people from there. Thus, whenever someone from off-site had a problem and did not know whom to contact at on-site, he would instead contact this liaison. This liaison was then able to work as a proxy between the sites as he knew most of the people from both sites. Team members from off-site were reported to be more eager to communicate towards on-site because of the liaison to whom they could speak in their native language. This also enabled a better trust between the sites, as team members from both sites were able to trust this person. However, in order to maintain the trust of both sites, it may be advisable for the liaison to be an active traveler.

Not only did the liaison help in establishing contacts between the sites, another benefit that was realized in Case Beta was that the liaison was able to work as a translator in the distributed meetings between the two sites. The developers from off-site were usually unable to understand everything that was said in English and were even sometimes afraid of speaking out load in English. The liaison was then able to work as an on-demand translator between the team members. Hence, a liaison that is able to fluently speak both sites's languages should be chosen.

A third benefit that was realized in Case Beta was that the work of on-site was becoming more transparent to the off-site team. The liaison was able to communicate to off-site about the true feeling of how the project was progressing at on-site. Below is a comment from a liaison that demonstrates the relationship between on-site and off-site:

> *"One time it looked like that [the customer] is going to cancel this project, and [the on-site project manager] was under big pressure, and he was a little bit maybe nervous and he was pushing [off-site]... They [off-site] had the feeling that they are going to be thrown away... And they kept having the feeling even after [the customer] decided to continue with the project... I was ensuring him [off-site project manager] that now "Hey, it is OK already, so you can trust that it's going to continue.""* - *Cultural liaison, Case Beta*

In Case Alpha, the project manager from on-site suggested that it might be more appropriate to transfer someone from the on-site team to the off-site team in the very beginning of the project. This person could then educate the off-site team about the working practices that are being used at on-site. If, however, one person from the off-site team would transfer to on-site, it would be harder for the rest of the off-site team to learn about the used working practices of on-site.

One drawback of designating a liaison is that it will introduce more costs for the project. It is not cheap to relocate one person for a long-term period, especially if he is to be an active traveler. Also, in two of the case projects that we studied, it was hard to find a person from on-site that was willing to stay at off-site for a long-term period. Careful and early planning is required in order to find a person who is willing to relocate himself for a longer time period.

### 4.3.5 Establishing efficient working practices

As distributed projects tend to be of reasonable size, it is important to understand that the processes and working practices cannot be ad-hoc, as they typically tend to be in some smaller co-located projects. Establishing working practices that will satisfy the needs of both sites will require time. The used practices should be discussed together with the whole or part of the team so that both sites can be committed to the used practices. After all, it should be in both sites's mutual interests to establish working practices that will allow everybody to work efficiently. Below is a comment from an off-site project manager that emphasizes the need of collaboratively establishing the working practices with all sites:

> *"So there were several times not enough communication to understand that processes, for example, were must in a distributed environment. It cannot be ad hoc. Otherwise we will fail in big problems." - Off-site project manager, Case Alpha*

Three of the studied case projects, namely Case Alpha, Case Beta and Case Gamma, used Rational Unified Process (RUP) as their software development process. As with most software processes, RUP is not a concrete process that will automatically fit the need of every project, but rather an adaptable process that should be tailored based on the needs of a project. Some of the studied case projects had realized this, while others had not. Careful discussion should take place when tailoring a standard software development process. Any variations from the standard should be clearly documented in order to avoid any ambiguities in later phases of the project. In Case Alpha, the off-site team members were too eager to implement RUP as-is, making the software development process too heavy. Below is a comment regarding this situation from the off-site project manager of this case:

> *"From the start there was a lot of misunderstanding because RUP is usually, or even let's say always, understood as something very rigid and very big, which at the end is not the reality. So it took us quite a long time to change the mindset and implement only those*

> *parts of RUP that were needed for our case." - Off-site project manager, Case Alpha*

For most distributed software projects, a standard software development process may not be suitable, as most software development processes have not been designed to work in distributed environments. Hence, special care should be taken to ensure the needs of all sites are fulfilled when tailoring the process.

### 4.3.6 Selecting tools and setting up infrastructure

The infrastructure at the off-site location should be set up directly after off-site has been involved into the project. This includes tasks such as:

- providing necessary hardware and software for the off-site team members,
- ensuring that the telecommunication links work,
- providing access to relevant project management systems, intranets, wikis and task management tools,
- setting up development and testing environments, and
- providing access to a common version management system.

In one of the studied case projects, namely Case Beta, proper care had not been taken in order to ensure that the off-site team could work efficiently with the used infrastructure. One of the most serious problem in this case was that all application and business data was considered very confidential by the case organization. The case organization was unwilling to allow the off-site team to replicate any of the data outside of the on-site team's servers. As a result, the off-site team had to work through a VPN connection at all times. This was seen as very frustrating by the off-site team members, as the VPN connection was lagging and disconnecting most of the time. Moreover, the off-site team became too dependent on the VPN connection. When there was a problem with the connection, the off-site team was unable to work.

Another major problem in this same case was the lack of an issue tracking tool. The off-site team had requested for an issue tracking tool which would have helped the project to track and manage its development tasks. However, this request was denied several times as on-site management did not want to save any confidential data in an issue tracking system.

> *"This is one of the most maybe painful issues that we still don't have issue tracking tool. So it's difficult to track what questions are not answered yet, and what's the status, and how many of them was, and how much time it took and so on." - Liaison, Case Beta*

In addition, the on-site management was not aware of the potential benefits that an issue tracking tool could provide. This is demonstrated in the following comment:

> *"Off-site wants to replace emails with an issue tracking tool, which is something that we [on-site] do not really understand. Email is already used for various things, so why should these be put in an issue tracking tool?" - On-site project owner, Case Beta*

Setting up an efficient working environment for the off-site team will require time. In most cases, choosing tools that are suitable for all sites requires a trial by error approach, i.e. trying out different tools till a suitable one is found.

### 4.3.7 Continuing by growing gradually

The previous sections have listed several practices and approaches that should be conducted or at least considered when initiating a new distributed software project. In order to be able to execute these practices, off-site's responsibilities should be increased gradually. This means that the tasks, activities and responsibilities of off-site should be increased in a step-by-step fashion. Especially, the development responsibilities of the product should be increased gradually. It is better to first solve the problems related to project management than to introduce new challenges related to product development. Hence, some time should be allocated to solve unforeseen problems that will eventually arise in the beginning of the project, as many distributed projects fail because of they do not invest enough time in a successful project setup.

To be able to do this, off-site's first task should be very a simple one, e.g. to complete a very limited version of the software product. This will allow the off-site team to simultaneously set up their working practices while working on their first relatively simple task. In addition, the off-site team is able to demonstrate that they are capable of working and delivering results with the used practices. Moreover, it is an efficient way to ensure that the off-site team has sufficient skills, resources and knowledge in order to work on the project.

> *"In the very first stages of the project, like the very first limited version of the software product which we had to deliver, and which we had to provide to the end customer ... it was really a chance for us to verify that we are able to work with this type of project and that we are able to deliver results, and so we are also able to get more responsibility." - Off-site production manager, Case Alpha*

After the off-site team has demonstrated a capability of working in the project and the project team has set up their working practices, more responsibilities could be transferred from on-site to off-site in a step-by-step fashion. Additional work can be moved to off-site when it is clear that they are working efficiently with their current set of responsibilities. In case there are problems with the current set of responsibilities, it is better to first solve these problems and then move more work from on-site to off-site. It is important to do this in small steps in order to avoid a chaotic environment.

Case Alpha used this approach. In this case, an impressive amount of work was gradually transferred from on-site to off-site. The first responsibilities that were moved from on-site to off-site were the testing of one product and the development of a very limited version of another product. This case project managed in the end to have around 80% of all the activities at the off-site location, whereas this percentage in the other case projects was somewhere around 40-50%. Eventually, this project continued to move responsibilities such as product architecture, requirements management and acceptance testing. This was possible because the on-site team had realized the amount of skills and competence available at the off-site location. However, this approach requires time and patience. In this case project, it took over 18 months for the on-site team to transfer most of their responsibilities to the off-site team. In the end, the amount of personnel at the on-site team was 5, whereas the number of people at the off-site team was 25.

There are several activities that should take place before the off-site can start working efficiently. Common examples of these are a joint kick-off meeting, knowledge transfer sessions, setting up the working practices and an environment that allows both on-site and off-site to work on the same code and artifacts. Moreover, it will take time to discuss and decide about the meeting, communication and collaboration practices. Below is a comment from the off-site project manager in Case Alpha, which demonstrates the amount of effort needed to stabilize the used working practices:

> *"We had quite long period when the processes were running from the one edge to the second edge, I would say, so they were changing quite a lot before they have been stabilizing, before we have agreed on a common way of working, a common way of communicating et cetera, and I would say it took us about half a year." - Off-site project manager, Case Alpha*

An interesting finding in Case Alpha was that moving responsibilities gradually from the on-site location to the off-site location increased the motivation of the off-site personnel to work on the project. Below is a comment from the off-site project manager of this project:

> *"Potentially start with bodyshopping, but move to service orientation with time. Okay, in the start probably it is better to assign very, very small task, but with time, continue with giving more responsibility to [off-site] personnel to motivate them by that." - Off-site project manager, Case Alpha*

As most of the activities in this case took place at the off-site location, the project became less distributed, which made the need for on-site and off-site to communicate and collaborate together much lower. Moving as much of the work as possible from on-site to off-site will naturally introduce some additional cost savings, as stated by the project owner of this project:

> *"Yeah, basically of course the basic reason is that the work [at off-site] is cheaper. The more we can put there, the more we save, if we can do it in a controlled and maneuverable, manageable way. And that's why we have in a way done it piece by piece. When we see that they are capable of taking more, then we give them more."*
> *- On-site project owner, Case Alpha*

### 4.3.8 Opening up the customer interface

In most global software development projects, including the case projects in this study, taking care of the customer relationship is usually the responsibility of the on-site team. Because of the potential cultural, time and language differences between the off-site team and the customer, there is usually not much collaboration between the two. Nevertheless, the communication channel between the off-site team and the customer was eventually opened in Case Alpha, which was the only case where this type of communication took place.

Direct communication between the off-site team and the customer was a very motivating factor for the off-site team. In the beginning of the project, all the communication was channeled through the on-site team. After more trust had been formed between the sites, the off-site team was able to participate in the meetings between the on-site team and the customer. First in a more passive role, just listening in, to be able to have the information directly from the customer. Eventually, the off-site team was allowed to communicate directly with the customer without the involvement of the on-site team. According to the on-site project manager, direct communication with the customer increases the off-site team's motivation and makes them more customer-oriented:

> *"It's very, very nice for the off-site project management to be more involved with the end customer. And when you are more involved with the end customer, we also saw from that point of view the change in behavior because they started to be in that sense much more customer-oriented. It feels much more deserved for that you can talk directly with the end customer. So that has also had a positive influence on the motivation of the team." - On-site project*

> *manager, Case Alpha*

Involving the off-site team in the customer meetings will also have other benefits than just increased motivation. Firstly, the communication becomes more clear as it does not have to be channeled through on-site. Secondly, the communication becomes more bidirectional as off-site is also able to comment on issues. In some meetings, the comments made by off-site were seen as very valuable, as noted by the on-site project manager:

> *"Gradually we have involved also off-site team members in those meetings. First in a more passive role, listening, having the information at first hand, so I [on-site project manager] don't need to tell [the off-site team members] everything again, with the risk that there are things not communicated, but also because of course the competence on off-site is getting better and better. They have a better understanding of the actual implementation than I do, so certain issues might not be possible at all, and so they can say it in the requirements meeting. So questions also are very valuable from the off-site unit, so it's not anymore a passive role. Today actually off-site has the requirements writing responsibility." - On-site project manager, Case Alpha*

The direct communication between the off-site team and the customer should be gradually increased. In the beginning, it may be enough to only involve the off-site project managers in the customer meetings. Once a trustful relationship has been established between the off-site team and the customer, more relevant off-site personnel may be involved in the customer meetings. In extreme cases, direct communication between the off-site team and the customer can also be allowed, without the involvement of the on-site team. Below is a comment from the on-site production manager:

> *"But I would say not the whole team, so just a couple of key persons being in the loop, and then gradually when you set up the operation or set up the project, when you have the different teams, then bring*

> *these people together with the relevant people on the customer side. Always have someone from on-site as much as possible also there in the meetings at least. I think that's, or in the beginning you should monitor that it goes fine. If you have the feeling it goes fine, that you're not needed so much anymore, you can skip, you can skip it sometimes. But I think still it's better that someone from on-site knows what's going on." - On-site project manager, Case Alpha*

Before involving the off-site team in the customer meetings, some training regarding communication etiquette may be advisable. In the first few customer meetings where off-site was involved, they way how some of the off-site team members expressed themselves was seen as quite rude by the on-site colleagues. The on-site project manager solved this problem by giving training regarding the communication etiquette and some selected phrases that the off-site team members could use in order to make their way of communicating more polite. Below is a comment from the project owner of this case that demonstrates how the off-site team members expressed themselves:

> *After the off-site team members were able to get into the customer interface, their way of communicating was quite rude. Sometimes they were communicating in a very rude way, something that a Finnish person wouldn't even say by mistake to the customer... Their attitude was that "Here is not everything that we need, we will not do this, forget about it"." - On-site project owner, Case Alpha*

## 4.4 Summary of findings

Table 4.2 presents a summary of the findings from the studied case projects.

Table 4.2: Summary of findings from the studied case projects

| PREPARING FOR A DISTRIBUTED PROJECT | |
|---|---|
| Evaluating the benefits of offshoring | • Evaluate if there is enough work for on-site team members after part of the work has been transferred to off-site.<br>• Ensure that all team members's communication, language, and working capabilities are suitable for a global environment.<br>• If an existing system is sent for offshore development, ensure that there is enough documentation in a language understood by the off-site team members. |
| Using cross-site teams over single-site teams | • Consider using cross-site teams in situations where the off-site team is heavily dependent on the knowledge of their on-site colleagues.<br>• A cross-site team structure forces frequent communication and collaboration between sites because of the introduced dependencies between team members.<br>• An increased collaboration between cross-site team members increases trust between sites. |
| Starting with system testing | • Consider allowing the off-site team to start with system testing when an existing system is sent for offshore development.<br>• Starting with system testing gives developers and increased know-how of how the system works. |

| | |
|---|---|
| Locating testing and development at the same site | • Consider locating testers and developers located at the same site whenever possible.<br>• Enables more frequent communication between testers and developers.<br>• Decreases the effort of transferring a testable product from development to testing.<br>• Allows the testing of new features to begin earlier. |
| Separating maintenance from development work | • Have a separate maintenance team for projects that have several incoming support requests or require customer-specific development.<br>• Locating the maintenance team at the same site where the customer is located, enables a quick response time to support requests.<br>• Allows the other development teams to completely focus on new feature development. |
| Structuring the teams and the organization | • Attempt to build a single project team instead of having sites clearly separated as on-site and off-site.<br>• A clear structure enables increased communication as team members will better understand everybody's responsibilities, and thus be able to know who to contact. |

| | |
|---|---|
| Motivating off-site with meaningful work | • Do not be resistant in transferring responsibilities to off-site.<br>• Giving out responsibilities to off-site in the beginning of a project is a good way of showing trust.<br>• Meaningful work and responsibilities are a good way to motivate the off-site team. |
| INITIATING A DISTRIBUTED PROJECT | |
| Recruiting new team members | • Attempt to recruit team members who have experience with global software projects and the used technologies.<br>• Attempt to recruit team members who have previously worked together. |
| Holding a face-to-face kick-off meeting | • Hold a face-to-face kick-off meeting together with the whole project team.<br>• A good opportunity to present background information about the project.<br>• Allows the team members to form personal ties with each others. |

| | |
|---|---|
| Training of new team members | • Provide both theoretical and practical training.<br>• Prepare enough training material for the training sessions.<br>• Provide training regarding business and product domain, and the used processes and tools.<br>• If there are huge cultural differences between sites, give training regarding these.<br>• Include team building activities in the training periods in order to build personal ties between team members<br>• Consider combining training sessions with the kick-off meeting to save on travel costs. |
| Designating liaisons | • Designating a liaison enables increased transparency and trust between the sites, as the liaison has the trust of both sites.<br>• The liaison should travel actively to maintain the trust of both sites.<br>• A liaison can establish contacts between sites and find relevant expertise in the project as he knows the project members from both sites.<br>• Plan the designation early, as it might be hard to find a person who is willing to relocate for a longer period of time. |
| Establishing efficient working practices | • Tailor the software process to fit the needs of the project.<br>• Collaboratively discuss and decide the used working practices. |

| | |
|---|---|
| Selecting tools and setting up infrastructure | • Prepare to invest a lot of effort when setting up the infrastructure.<br>• Provide a task management tool to increase the transparency between sites. |
| Continuing by growing gradually | • Allow off-site to start working with a limited version of the product during the ramp-up period.<br>• Increase off-site's responsibilities in a step-by-step fashion once they have proved they are capable of taking more responsibilities.<br>• The off-site team can be kept motivated by continuously giving more meaningful work. |
| Opening up the customer interface | • Off-site team members should be able to also communicate with the customer representatives. Improves communication and the understanding of customer needs.<br>• Consider giving off-site team members training regarding communication etiquette. |

# Chapter 5

# Discussion and Conclusions

This final chapter presents the conclusions of this study. The chapter begins with section 5.1 that presents the answers to the two research questions. The chapter continues with a discussion and comparison of results and literature in section 5.2. After this, the limitations that effected this study are introduced in section 5.3. Finally, the last section 5.4 concludes the chapter with ideas regarding what future work should focus on.

## 5.1 Answers to the research questions

This section presents the answers to the research questions. The overall research problem was:

*What should be done in the early-life of a distributed software development project in order for it to have a better chance of success?*

This research problem was further divided into two research questions in section 2.1. The next two sections present the answers to the research questions.

### 5.1.1 Preparing for a distributed project

**Research Question 1**: *What factors should be taken into account when preparing for a distributed software development project?*

Based on the studied case projects, a multitude of issues need to be considered when preparing for a GSD project. These range from establishing a plan on what to distribute, to different approaches for structuring the teams and the organization.

Based on the results from the studied case projects, it was possible to identify several successful approaches for structuring the teams and the organization, some of which may only work for certain situations and contexts. First, it may be ideal to separate the development of new functionalities from maintenance work for projects that have several incoming support requests or require customer-specific development. Locating the maintenance teams at the sites where the customers are located enables a better collaboration between the project and its customers. In addition, the development teams are able to completely focus on new feature development, as the maintenance teams are taking care of the daily support requests. Secondly, a cross-site team structure should be used over a single-site team structure for situations where the off-site team is heavily dependent on the knowledge of their on-site colleagues. This forces a more frequent communication and collaboration between sites as the team members from different sites become more dependent on one another. As a third approach, for projects where an existing system is sent for offshore development, the off-site team should be allowed to start with system testing. If the first task of the off-site team members is to test the functionality of the system, they are able to get an increased know-how of how the system works before they start developing it. A final fourth finding regarding team structuring is to locate testers and developers at the same site. This introduces benefits such as a decreased effort in transferring a testable product from development to testing, increased communication as testers and developers are able to communicate face-to-face, and it allowed the testing of new features to begin earlier.

Issues regarding trust should be addressed as early-on as possible. After the team and organization structures have been decided, it is vital that this structure is clear and it is efficiently communicated to everybody. From the very beginning of the project, everybody's role in the project should be emphasized and an atmosphere of partnership should be created by involving team members from both sites in the decision making processes. One good way of building a trustful relationship between the sites is to give meaningful work to the off-site teams. The organization should not be resistant in transferring meaningful responsibilities to off-site. Engaging the off-site team members with meaningful work has a high effect on their motivation and it helps in retaining the best talent at off-site.

An important finding was to evaluate the maturity of the project before preparing for distribution. It should be ensured that the existing on-site team members have good communication, language and working capabilities to work in a global environment. Difficulties arised especially in situations where both on-site and off-site were communicating with a foreign language. Moreover, if the existing documentation is in a language not understood by the off-site team members, the off-site team members may have several difficulties in understanding the product and business domain. Therefore, if the project involves an existing system, it should be ensured that there is enough documentation in a language understood by the off-site team members.

### 5.1.2 Initiating a distributed project

**Research Question 2**: *What practices should be implemented when initiating a distributed software development project?*

Based on the findings from the studied case projects, there are several practices that should be implemented directly after the initiation of a distributed software project in order for it to have a better chance of success. These range from holding a face-to-face kick-off meeting, training new team members, establishing efficient working practices, to designating liaisons.

Based on the results from the studied case projects, allowing the project team members to form personal ties with one another was seen as cruicial. Thus, the project should be initiated with a face-to-face kick-off meeting together with the whole project team. This is a good opportunity to present relevant project background information so that everybody gets a common understanding of what is to be done during the project. As the kick-off meeting is the first opportunity for project members to meet each others, there should be a lot of time allocated for social activities to encourage integration on a social level and the formation of personal relationships. The social activities allowed the team members from different sites to form personal ties to one another, making communication much more easier once the project members were distributed.

Training of team members was considered to be one of the most important activities in the beginning of the project by the interviewees. Training regarding cultural differences, the used process, the used technologies, and the product and business domain was seen as important. As providing both training and knowledge transfer can be expensive because they effectively use the traveling budget and both on-site's and off-site's human resources, careful planning regarding the contents, timing and who gets to participate was seen as important.

During the ramp-up phase of a project, there is often a growing need for new recruits. During the recruitment process, one should try to recruit team members who have previous experiences with global software development projects and the used technologies. Moreover, recruiting team members who have previously worked together helps building trust within the team. This can be done by asking qualified recruits about people whom they have worked with in the past. Along with the recruitment of new team members, the project will also need to appoint an experienced leader. This person should have previous experience in global collaborations, strong communication skills, and exposure to different cultures.

Existing software development process models may not fit distributed environments as such. Therefore, the used process model should be tailored to fit the needs of all involved sites. This requires that the working practices are collaboratively discussed and decided so that a common understanding of how work should be performed can be achieved. For example, it must be ensured that the selected tools, technologies and infrastructure is accessible and compatible across the sites.

Several interviewees noted that designating a liaison increases the transparency across sites as the liaison knows the environment and people at both sites. In addition to this, the liaison is able to facilitate communication by establishing contacts between sites. Despite this, many of the studied case projects did not designate a liaison as there was no-one who was willing to relocate for a longer period of time.

## 5.2 Comparison between results and literature

### 5.2.1 Preparing for a distributed project

Before initiating a globally distributed software project, careful evaluation should take place. Lings *et al.* (2007) suggest to choose offshore teams that have a common language with the on-site teams. According to the results from the studied case projects, it should be evaluated if all team members's communication, language, and working capabilities are suitable for a global environment. The literature continues to suggest to evaluate several key characteristics of a project, such as the business process, complexity, current cost, and risk of failure (Cusick & Prasad, 2006).

The importance of advance planning and preparation in the beginning of a distributed project is widely emphasized in the literature (Cusick & Prasad, 2006; Prikladnicki *et al.* , 2003; Mettovaara *et al.* , 2006). A planning phase allows the project to get a organized and managed ramp-up (Prikladnicki *et al.* , 2003). The literature suggests that a team and project vision should be built collaboratively (Bhat *et al.* , 2006). This was severely lacking in some of the studied case projects, with the ramp-up phase being often described as chaotic and unorganized.

Findings regarding the benefits of a cross-site team structure differed between the literature and results. Cross-site teams with team members from different countries can be useful to integrate the different cultures (Ebert *et al.* , 2001). The findings from the studied case projects differed by suggesting benefits such as forced and frequent communication between team members, increased collaboration between the sites, and increased trust between the team members. In addition, the introduction of cross-site teams allowed the teams to have common goals and a shared vision, which made the team members more committed to the project.

Allowing testers and developers to work together was also noted by both the literature and the results. This type of team structuring impacts team collaboration and helps anticipating defects early on (Cristal *et al.* , 2008). According to the studied case projects, other benefits included a decreased effort in transferring a testable product from development to testing, enhanced communication between testers and developers, and a potential increase in product quality.

Another interesting finding which could not be found from the literature was to consider allowing the off-site team members to begin their work with system testing. This applies to situations where an existing system is sent for offshore development. If the first task of the off-site team members is to test the functionality of the system, they are able to get an increased know-how of how the system works before they start developing it.

The literature widely mentions about the importance of taking a long-term strategic view (Stark *et al.* , 2006; Khan *et al.* , 2009; Tervonen & Mustonen, 2009), as a continuous cooperation will provide both parties increased benefits as time passes by. The more successful case projects had also realized the importance of taking a long-term view by investing heavily in the ramp-up period.

### 5.2.2 Initiating a distributed project

According to the results and the literature, holding a face-to-face kick-off meeting in the beginning of the project was considered as very important (Pyysiäinen, 2003; Berenbach, 2006; Raffo & Setamanit, 2005; Bhat *et al.* , 2006; Mettovaara *et al.* , 2006). The importance of involving social activities was noted by both the results and the literature. Social activities can be used to encourage integration on a social level (Gotel *et al.* , 2008).

Differences and similarities regarding the content of training existed in the results and the literature. According to the interviewees from the studied case projects, training regarding cultural differences, the used process, the used technologies, and the product and business domain was seen as important. Even though many interviewees saw these topics as important, most projects focused mainly on providing training regarding the used technologies and knowledge transfer regarding the product domain. Literature further suggests that training should be provided regarding topics such as communication, tools, technologies (Bhat *et al.* , 2006), culture, language (Lings *et al.* , 2007), leadership, context sharing, project management (Prikladnicki *et al.* , 2003) and the used development process (Lings *et al.* , 2006). In addition, the training should consist of project-specific norms for meetings, deadlines and commitments (Paré & Dubé, 1999).

Careful selection of new recruits in order to get talented resources was seen as important by both the literature and the results. According to Hofner *et al.* (2007) and many interviewees from the case projects, the potential recruits should be interviewed, especially the senior project members. Despite this, several case projects did not engage in interviewing the potential recruits. The continuity of the off-site staff was also regarded as important (Cusick & Prasad, 2006). Differences regarding the approaches on how to achieve this varied between the results and the literature. In one case project, meaningful work and responsibilities together with direct contact with the customer representatives was seen as motivating for the off-site team members. According to literature, providing good career advancement opportunities (Jensen *et al.* , 2007), high-end work with advanced technologies (Kobitzsch *et al.* , 2001), and effective team integration with strong leadership (Hofner *et al.* , 2007) are good approaches for ensuring the continuity of the off-site staff.

Selecting tools and setting up infrastructure was seen as another important activity that should take place in the beginning of the project. According to the interviewees and Dubé & Paré (2001), time and effort should be spent on ensuring that the appropriate tools, technologies and infrastructures are accessible and compatible across the sites. According to the findings from the studied case projects, much time is especially required for finding and selecting tools that are suitable for all sites, as this often requires a trial by error approach, i.e. trying out different tools till a suitable one is found. Moreover, Cusick & Prasad (2006) note that time is needed for mirroring a common infrastructure environment at both sites.

The results coincided with the literature regarding the benefits and challenges of designating liaisons, with the main benefits being a better integration of the distributed teams (Bass *et al.* , 2007) and an increased awareness of the other sites's culture (Lings *et al.* , 2007). The main challenge was in finding a good candidate who was willing to relocate himself for a longer time period (Kobitzsch *et al.* , 2001).

## 5.3 Limitations

A number of limitations have effected this study. First of all, the small number of case projects included in this study may not offer grounds for generalizability of the findings. Another threat regarding generalizability is that the study was conducted within the boundaries of a single case organization. The case organization consisted of several sub-organizations, as its strategy was to gain growth by setting up offshoring centers in countries where talented and cheap labor existed. Even though the organization's size and global presence does create a multi-organizational diversity, the findings may still not be applicable in all organizational contexts. In this case organization and the studied case projects, much of the existing work was being transferred from on-site to the offshoring centers. Thus, some of the findings may only be applicable to projects where existing work is transferred for offshore development.

The interviews themselves may have also posed some limitations. For example, the interview templates (see appendix A) contained only one section related to the ramp-up phase of the project. Moreover, in three of the cases, the interviews with on-site team members were conducted in Finnish, whereas the interviews with off-site team members were always conducted in English. Finnish was a mother tongue of both the interviewers and the interviewees, whereas English was not the mother tongue of neither the interviewers nor the interviewees. This may have also posed another threat, since the on-site team members in these three case projects had often more to say than their off-site colleagues. In a few interviews, the off-site team members had a small difficulty of expressing themselves clearly.

## 5.4 Future work

Future work is required in order to more clearly understand how globally distributed software projects can be initiated in a cost-effective fashion. Implementing all of the listed practices is often unreasonable and costly. Moreover, even though one practice can suit a specific organizational or project context, it may not be suitable in other contexts. As such, there needs to be a straightforward way of determining which practices to apply for a certain context. In addition, many of the identified practices are addressing similar problems. Therefore, a framework for choosing suitable practices for a specific context could be useful. This framework should also include some type of mechanism to monitor the effectiveness of the chosen practices.

# REFERENCES

Bass M., Herbsleb JD, & Lescher C. 2007. Collaboration in Global Software Projects at Siemens: An Experience Report. *Pages 33–39 of: Second IEEE International Conference on Global Software Engineering, 2007. ICGSE 2007*. IEEE Computer Society.

Battin R.D., Crocker R., Kreidler J., & Subramanian K. 2001. Leveraging Resources in Global Software Development. *IEEE Software*, **18**(2), 70–77.

Berenbach B. 2006. Impact of Organizational Structure on Distributed Requirements Engineering Processes: Lessons Learned. *Pages 15–19 of: International Conference on Software Engineering: Proceedings of the 2006 international workshop on Global software development for the practitioner*.

Bhat J.M., Gupta M., & Murthy S.N. 2006. Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing. *IEEE Software*, **23**(5), 38–44.

Bondi A.B., & Ros J.P. 2009. Experience with Training a Remotely Located Performance Test Team in a Quasi-Agile Global Environment. *Pages 254–261 of: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*. IEEE Computer Society.

Carmel E. 1999. *Global Software Teams: Collaborating Across Borders and Time Zones*. Prentice Hall.

Carmel E., & Agarwal R. 2001. Tactical Approaches for Alleviating Distance in Global Software Development. *IEEE Software*, **18**(2), 22–29.

Casey V., & Richardson I. 2006. Project Management within Virtual Software Teams. *Pages 33–42 of: Proceedings of the IEEE international conference on Global Software Engineering*. IEEE Computer Society.

Cataldo M., & Nambiar S. 2009. Quality in Global Software Development Projects: A Closer Look at the Role of Distribution. *Pages 163–172 of: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*. IEEE Computer Society.

Conchuir E.O., Holmstrom H., Agerfalk P.J., & Fitzgerald B. 2006. Exploring the Assumed Benefits of Global Software Development. *Pages 159–168 of: Proceedings of the IEEE international conference on Global Software Engineering*. IEEE Computer Society.

Cristal M., Wildt D., & Prikladnicki R. 2008. Usage of SCRUM Practices within a Global Company. *Pages 222–226 of: Proceedings of the 2008 IEEE International Conference on Global Software Engineering*. IEEE Computer Society.

Cusick J., & Prasad A. 2006. A Practical Management and Engineering Approach to Offshore Collaboration. *IEEE Software*, **23**(5), 20–29.

Damian D. 2008. *Beyond travel: Unleashing the collaborative potential of global teams - Guidelines for successful distributed collaboration*. Tech. rept. University of Victoria.

Dubé L., & Paré G. 2001. Global Virtual Teams. *Communications of the ACM*, **44**(12), 71–73.

Ebert C., De Neve P., & Alcatel A. 2001. Surviving Global Software Development. *IEEE Software*, **18**(2), 62–69.

Erenkrantz J.R., & Taylor R.N. 2003. Supporting Distributed and Decentralized Projects: Drawing Lessons from the Open Source Community. *In: Proceedings of 1st Workshop on Open Source in an Industrial Context*.

Forbath T., Brooks P., & Dass A. 2008. Beyond Cost Reduction: Using Collaboration to Increase Innovation in Global Software Development Projects. *Pages 205–209 of: IEEE International Conference on Global Software Engineering, 2008. ICGSE 2008*. IEEE Computer Society.

Gotel O., Kulkarni V., Scharff C., & Neak L. 2008. Integration starts on day one in global software development projects. *Pages 244–248 of: IEEE International Conference on Global Software Engineering, 2008. ICGSE 2008*. IEEE Computer Society.

Grinter R.E., Herbsleb J.D., & Perry D.E. 1999. The geography of coordination: dealing with distance in R&D work. *Pages 306–315 of: Proceedings of the international ACM SIGGROUP conference on Supporting Group Work*. ACM.

Heeks R., Krishna S., Nicholsen B., & Sahay S. 2001. Synching or Sinking: Global Software Outsourcing Relationships. *IEEE software*, **18**(2), 54–60.

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

Herbsleb J.D., & Mockus A. 2003. An Empirical Study of Speed and Communication in Globally-Distributed Software Development. *IEEE Transactions on Software Engineering*, **29**(6), 481–494.

Herbsleb J.D., & Moitra D. 2001. Global Software Development. *IEEE Software*, **18**(2), 16–20.

Herbsleb J.D., Paulish D.J., & Bass M. 2005. Global Software Development at Siemens: Experience from Nine Projects. *Pages 524–533 of: Proceedings of the 27th international conference on Software engineering*.

Hofner G., Mani VS, Ltd S.I.S., & Delhi N. 2007. TAPER: A generic framework for establishing an offshore development center. *Pages 162–172 of: Second IEEE International Conference on Global Software Engineering, 2007. ICGSE 2007*. IEEE Computer Society.

Jensen M., Menon S., Mangset L.E., & Dalberg V. 2007. Managing Offshore Outsourcing of Knowledge-intensive Projects-A People Centric Approach. *Pages 186–196 of: Proceedings of the International Conference on Global Software Engineering*. IEEE Computer Society.

Khan S.U., Niazi M., & Ahmad R. 2009. Critical Success Factors for Offshore Software Development Outsourcing Vendors: A Systematic Literature Review. *Pages 207–216 of: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*. IEEE Computer Society.

Kobitzsch W., Rombach D., & Feldmann R.L. 2001. Outsourcing in India. *IEEE Software*, **18**(2), 78–86.

Komi-Sirviö S., & Tihinen M. 2005. Lessons Learned by Participants of Distributed Software Development. *Knowledge and Process Management*, **12**(2), 108–122.

Lamersdorf A., M
"unch J., & Rombach D. 2008. Towards a Multi-Criteria Development Distribution Model: An Analysis of Existing Task Distribution Approaches. *Pages 109–118 of: Proceedings of the 2008 IEEE International Conference on Global Software Engineering*. IEEE Computer Society.

Leszak M., Meier M., & Alcatel-Lucent N. 2007. Successful Global Development of a Large-scale Embedded Telecommunications Product. *Pages 23–32 of: Second IEEE International Conference on Global Software Engineering, 2007. ICGSE 2007*. IEEE Computer Society.

Lindqvist E., Lundell B., & Lings B. 2006. Distributed Development in an Intra-national, Intra-organisational Context: An Experience Report. *Pages 80–86 of: International Conference on Software Engineering: Proceedings of the 2006 international workshop on Global software development for the practitioner*.

Lings B., Lundell B., Ågerfalk P., & Fitzgerald B. 2006. Ten Strategies for Successful Distributed Development. *International Federation for Infor-*

*mation Processing (IFIP): The Transfer and Diffusion of Information Technology for Organizational Resilience*, **206**, 119–137.

Lings B., Lundell B., Agerfalk PJ, & Fitzgerald B. 2007. A reference model for successful Distributed Development of Software Systems. *Pages 130–139 of: Second IEEE International Conference on Global Software Engineering, 2007. ICGSE 2007.* IEEE Computer Society.

Mettovaara V., Siponen M.T., & Lehto J.A. 2006. Collaboration in Software Development: Lesson Learned from Two Large Multinational Organizations. *PACIS 2006 Proceedings*, 103.

Mockus A., & Weiss DM. 2001. Globalization by Chunking: A Quantitative Approach. *IEEE Software*, **18**(2), 30–37.

Mohan S., & Fernandez J. 2010. Distributed Software Development Projects: Work Breakdown Approaches to Overcome Key Coordination Challenges. *Pages 173–182 of: Proceedings of the 3rd India Software Engineering Conference.*

Narayanan S., Mazumder S., *et al.* . 2006. Success of Offshore Relationships: Engineering team structures. *Pages 73–82 of: Proceedings of the IEEE international conference on Global Software Engineering.* IEEE Computer Society.

Paasivaara M. 2005. *Communication Practices in Inter-organisational Product Development.* Ph.D. thesis, Helsinki University of Technology.

Paré G., & Dubé L. 1999. Virtual Teams: An Exploratory Study of Key Challenges and Strategies. *Pages 479–483 of: Proceedings of the 20th international conference on Information Systems.* ACM.

Prikladnicki R., & Pilatti L. 2008. Improving Contextual Skills in Global Software Engineering: A Corporate Training Experience. *Pages 239–243 of: IEEE International Conference on Global Software Engineering, 2008. ICGSE 2008.* IEEE Computer Society.

Prikladnicki R., Audy J.L.N., & Evaristo R. 2003. Global Software Development in Practice Lessons Learned. *Software Process: Improvement and Practice*, **8**(4), 267–281.

Pyysiäinen J. 2003. Building Trust in Global Inter-Organizational Software Development Projects: Problems and Practices. *Page 69 of: GSD'03 The International Workshop on Global Software Development.*

Raffo D., & Setamanit S. 2005. A Simulation Model for Global Software Development Project. *In: The International Workshop on Software Process Simulation and Modeling.*

Stark J., Arlt M., & Walker D.H.T. 2006. Outsourcing Decisions and Models-Some Practical Considerations for Large Organizations. *Pages 12–17 of: Proceedings of the IEEE international conference on Global Software Engineering.* IEEE Computer Society.

Sudershana S., Villca-Roque A., & Baldanza J. 2007. Successful Collaborative Software Projects for Medical Devices in an FDA Regulated Environment: Myth or Reality? *Pages 217–224 of: Second IEEE International Conference on Global Software Engineering, 2007. ICGSE 2007.* IEEE Computer Society.

Sureshchandra K., & Shrinivasavadhani J. 2008. Adopting Agile in Distributed Development. *Pages 217–221 of: IEEE International Conference on Global Software Engineering, 2008. ICGSE 2008.* IEEE Computer Society.

Tellis W. 1997. Introduction to Case Study. *The Qualitative Report*, **3**(2), 1–11.

Tervonen I., & Mustonen T. 2009. Offshoring Test Automation: Observations and Lessons Learned. *Pages 226–235 of: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering.* IEEE Computer Society.

Yin Robert K. 1994. *Case study research: Design and Methods.* 2nd edn. Applied Social Research Methods Series, vol. 5. Sage Publications, Inc.

# Appendix A

# Interview templates

## A.1 Template for Case Alpha, Case Beta and Case Gamma

The following is a questionnaire template for semi-structured interviews held in Case Alpha, Case Beta and Case Gamma.

**1. Interviewee**

- Background (education, experience)
- Role and tasks in the project
- When did you start in current role / this project
- Expectations for our project / this research

**2. Collaboration Background**

- Length and type of collaboration
- Reasons for collaboration
- Initiation of collaboration relationship
- Vision of future collaboration

**3. Project Background**

- Purpose of collaboration
- Parties, roles, teams (number of persons), project managers
- Why were these parties chosen?
- Project type – uncertainty of environment / requirements
- Contract type
- Project plan (was it done / what included)

- Schedule – planned / realized

4. **Beginning of the project**

- How did it start?
- Work division
- Requirement specification (who participated)
- Kick-off / meetings /training

5. **Process model**

- Iterations
- Delivery cycle, delivery chain
- Feedback to deliveries - quality
- Why was this process model chosen – training/understanding
- Synchronization
- Integration, frequency of builds,
- Testing (who/when)
- Problems/benefits

6. **Data management**

- Versions, configurations management
- Common repository between sites?
- Change management

7. **Monitoring**

- How is project progress followed / by whom
- Transparency of the project/progress to team members - informing

8. **Communication**

- Communicating roles / link persons /peer-to-peer links
- Meetings
- Informal communication
- Media: email / phone /face-to-face (when / why /with whom)
- Chat / messenger usage
- Which communication tool do you prefer? Why?
- Problem solving: how / responsible
- How large part of your working time is used on communication?
- Major communication problems
- Improvement ideas

**9. Trust & feedback**

- Do you feel you can trust the distributed colleagues/teams?
- Getting help / how do you get answers
- Making suggestions and getting response
- Do you know enough people from other sites / well enough
- Do you feel that you are one team, or several teams (between sites)?
- Misunderstandings (why / when)
- Style of communication
- Openness – how easily are problems communicated / contacts initiated
- Do you get enough information (for your work / about the whole project)
- Do you get feedback on your work

**10. Distance**

- Cultural differences
- Geographical distance
- Time difference
- Power distance / hierarchical – egalitarian

**11. End of project**

- Lessons learned

**12. Problems / successes (mention 3 the most important ones)**

- What were the main problems in the collaboration
- Improvement ideas
- Successful practices
- Challenges / benefits of distributed work compared to traditional collocated work

## A.2 Template for Case Delta

The following is a questionnaire template for semi-structured interviews held in Case Delta.

**1. Interviewee**

- What is your education?
- When did you join the project (months; phase of the project)?
- What are your main responsibilities (tasks)?

- How long have you been doing the current tasks?
- Experience with any agile methods or distributed projects before?
- Are you involved in other projects at the same time when working in this project? How much time do you spend on this project? Give a percentage.

**2. Project Background**

- Purpose of the project and the collaboration?
- How many sites are there (in which countries)?
- Number of customer locations?
- How many people and teams are there?
- How do you manage your people?
- How does the team manage itself?
- What roles exist in the project?
- Are there descriptions of roles?
- Project duration and start
- Schedule (planned and realized)
- How is the work organized and divided between sites?
- Is the development work split into several modules?
- Has the chosen division caused any problems?

**3. Process model**

- What process model is used in this project?
- Do you have a description of the process?
- Why was this process model chosen (training, understanding)?
- Do you have milestones? How often?
- Process improvement? Do you reflect regularly on the process?

**4. Agile methods**

- Describe the agile methods you are using?
- Which practices do you use?
- Which practices where tried but dismissed and why?
- Why did you not take some other practices into use?
- Which practices required adaption?
- What new practices were developed?
- What practices could be improved?

**5. Releases and iterations**

- How do you prioritize requirements? Especially at the beginning of the project?

- Describe a release?
- Describe an iteration?
  - How long are the iteration cycles?
  - Are the cycles of the same size for all sites? (Synchronization)
- What kind of feedback do you get to deliveries?

**6. Data management**

- How do you take care of version control?
- How are requirements documented?
- What else is documented?
- Is there a central repository available for every member 247? Also across sites?
- How is the project, a release, an iteration monitored?
- How do you ensure that everyone gets proper information on project status?
- Where is the backlog located?
- How do you take care of configurations management?

**7. Testing**

- How is testing arranged?
- How often is testing performed?
- Who does the testing?

**8. Communication**

- What kind of meetings do you have between sites?
  - Frequency?
  - Contents?
  - Participants?
  - Face-to-face?
  - During project start up?
  - During the project?
  - How many do you think are necessary?
  - Can some be replaced by other practices? If yes, through which?
- What media do you choose for these meetings? Which one is the most important one?
- Are there main roles that have communication responsibilities?
- In what other ways do you discuss project matters across sites?
- How do people communicate with each other on daily basis?

- Did you agree on response times for certain communication (email, missed calls, forwarded information)?
- What are the major communication difficulties? Improvement ideas? Successful practices?
- Are agile practices used explicitly to enhance communication? If yes, which?

**9. Time zone differences**

- Is there a difference in time zones?
- Does that cause any problems?
- Do you synchronize your work? (same meeting hours, one stays longer and one comes earlier)?
- Who is responsible for synchronization?

**10. Traveling**

- Short trips
  - Do you have short trips to the other site?
  - How often?
  - For what purpose?
  - Who travels?
- Long trips
  - How often do you have long trips?
  - Who travels?
  - How long are the journeys?
  - For what purpose?
  - How much collocated work is required in a distributed agile project in your opinion?
  - How can collocated work be decreased through other (agile) practices?

**11. People Management:**

- How are the customers involved?
- How does it affect on the work of developers?
- How does the team manage itself?

**12. Familiarity**

- Do you feel that you have one team or separate teams?

- Where do you feel that you belong to?
- What is done to enhance team culture?
- How many people from other sites do you know? How much contact do you have with them? Do you trust them?
- Have you ever met them in person?
- Is it easy to get help? How do you get answers to your questions?
- Have there been misunderstandings?
- How do you think misunderstandings can be avoided?
- How do you get feedback on your work? Would you need some more?

**13. Problems**

- What kind of disagreements does your project have with partner companies?

**14. End**

- Which practices are the most useful ones and why?
- Is there something you would like to improve?
- What where the three biggest problems with agile practices?
- Do agile methods suit for distributed settings? Why? Why not?
- What advice would you give to other companies?

# Appendix B

# Template for thematic coding

- Introduction
  - Interviewee background
  - Involved sites
  - Customers
  - Product
  - Organization structure
  - Timeline of events
- Process
  - Task management
  - Division of work
- Used practices
- Meetings
- Traveling
- Support systems
- Differences between sites
  - Geographical differences
  - Cultural differences
- Pros & Cons
- Lessons learned
- Worthy quotes