PRACTICAL GUIDE TO MANAGING DISTRIBUTED SOFTWARE DEVELOPMENT PROJECTS

Maria Paasivaara   Nico Hiort af Ornäs   Peitsa Hynninen
Casper Lassenius   Tuomas Niinimäki   Arttu Piri

Aalto University School of Science and Technology
Department of Computer Science and Engineering
Software Business and Engineering Institute

Aalto-yliopiston teknillinen korkeakoulu
Tietotekniikan laitos
Ohjelmistoliiketoiminnan ja –tuotannon laboratorio
Contents

I Planning and Practices

1 Initiating Global Software Development 3

1.1 Introduction .................................................. 3

1.2 Setting up the collaboration .................................. 3

1.2.1 Communicate the overall GSD strategy .................. 3

1.2.2 Division of work ........................................ 5

1.3 Starting a GSD Project ...................................... 8

1.3.1 Arrange a project kick-off .............................. 8

1.3.2 Invest in knowledge transfer ............................ 9

1.3.3 Increase off-site responsibilities gradually ............ 11

1.3.4 Involve the whole team in planning ............. 12

1.3.5 Implement good working practices and processes ...... 13

1.3.6 Designate liaisons ...................................... 14

2 Communication and Collaboration Practices 17

2.1 Introduction .................................................. 17

2.2 Challenges ................................................... 17

2.3 Practices .................................................... 19

2.3.1 Use frequent deliveries ............................... 19

2.3.2 Reduce temporal distance ............................... 20

2.3.3 Reduce the number of sites ............................ 21

2.3.4 Reduce cultural distance ............................... 21

2.3.5 Define and communicate a clear organizational structure .... 22

2.3.6 Prefer dedicated team members ....................... 23

2.3.7 Create one single team ................................. 23

2.3.8 Arrange face-to-face team building possibilities .... 24

2.3.9 Provide background information ..................... 24

2.3.10 Agree on communication practices and tools ........ 24

2.3.11 Agree on progress monitoring ....................... 26

2.3.12 Arrange training ....................................... 26

2.3.13 Create visibility across sites ....................... 27

2.3.14 Arrange frequent status meetings .................. 27

2.3.15 Arrange weekly meetings across teams ............. 28

2.3.16 Encourage frequent communication ................ 28
## III Tools and Infrastructure

### 9 Choosing Communication Media for GSD

9.1 Introduction ........................................ 111  
9.2 Fundamentals of tool support for communication  
9.2.1 Media richness theory ............................. 112  
9.2.2 Media Synchronicity Theory ..................... 114  
9.3 Properties of communication and collaboration tools  
9.3.1 Immediacy of feedback ........................... 116  
9.3.2 Symbol variety .................................... 116  
9.3.3 Parallellism ....................................... 118  
9.3.4 Rehearsability ................................... 118  
9.3.5 Reprocessability .................................. 119  
9.4 Requirements for collaboration tools in GSD .......... 119  
9.4.1 Geographical distance ............................ 119  
9.4.2 Temporal distance ................................. 120  
9.4.3 Organizational distance ......................... 121  
9.4.4 Cultural distance ................................ 123  
9.5 Professional role in media choice .................... 124  
9.6 Communicating technical information ................ 125  
9.7 Communication modes ............................... 125  
9.8 Summary ............................................. 125

### 10 Communication Tools for GSD projects

10.1 Introduction ......................................... 129  
10.2 Synchronous communication .......................... 129  
10.2.1 Instant messaging ................................ 130  
10.2.2 Teleconferencing ................................ 134  
10.2.3 Telephones ....................................... 136  
10.2.4 Video conferencing ............................... 137  
10.2.5 Desktop sharing .................................. 139  
10.3 Asynchronous communication ........................ 140  
10.3.1 Email ............................................. 141  
10.3.2 Issue trackers .................................... 143  
10.4 Summary ............................................. 145

### 11 Physical Workspaces and Distributed Teams

11.1 Introduction ......................................... 149  
11.2 Challenges .......................................... 150  
11.3 Solutions ............................................ 152

## Appendix: Research Methods

A.1 Research project and goals of the study ............ 157  
A.2 Multiple-case study ................................... 157  
A.3 Data collection ....................................... 158
A.4 Data analysis
Preface

Introduction

This book is aimed at practitioners working with or in globally distributed software development projects: project managers, developers, architects, testers, and other project team members. In this book, we provide practical advice for "surviving" the difficult waters of global software development. The advice is based upon the experiences of companies, collected by us as researchers in the MaPIT (Management, Processes and IT Support for Globally Distributed Software Development) research project, as well as on our earlier experiences while working with companies introducing or using global software engineering.

Global Software Development — Why?

Global software development (GSD) is becoming increasingly common among software development companies. Many companies have at least a few projects involving remote team members or partners, scattered either across a country or the globe. Running such distributed projects is, however, never an easy undertaking. Instead, it includes additional challenges that a collocated project does not have to deal with. Despite the challenges, there are several reasons for companies to use global software development. Some of the most common ones include:

- **Lower cost.** Software companies want to exploit the wage differences between the developed economies and lower wage countries, such as Asian, South American or Eastern European countries. At the moment, this seems to be the single most important driver for companies to establish own offsite development centers or to use foreign subcontractors.

- **Mergers and acquisitions.** Software companies often grow by buying other companies or by merging with partners or competitors. This
leads to companies that are internally distributed. Different sites can remain quite independent, and collaboration across sites can be even more challenging than between legally separate companies.

- **Development closer to the market / customers.** The markets for software companies are increasingly global. Companies might establish sales offices or local development centers in the countries where their main customers are located, to localize software to suit local markets and regulations or to collect information on local needs and requirements.

- **Increasing the speed of development.** Companies need to bring new products into markets very quickly. When own resources in the home country are scarce, hiring a partner or subcontractor, or establishing an offsite development center that can do development work in parallel with the home office, can increase the development speed.

- **Concentration on core competences.** Companies are increasingly concentrating on their core competences, i.e., they do only what they can do best and buy everything else from partners and subcontractors.

- **Lack of own resources.** Companies growing fast might find it difficult to hire competent resources quickly enough in their current locations. Thus, hiring a subcontractor or a partner might be a faster and safer alternative. If the market situation turns out to be bad, it is easier to downsize subcontractor resources rather than the own workforce. Even an own offsite center might offer similar advantages.

- **Lack of knowledge.** Many companies concentrating on core competences find it easier and faster to hire a partner or a subcontractor when new special knowledge is needed. Developing own new knowledge from scratch might take too long and the market opportunity might be lost.

The list above presented some of the reasons for software companies to have globally distributed projects. It seems that in the future the globally distributed software development projects are the norm. However, successfully managing distributed projects is more difficult than companies often expect. It is common for distributed projects to take longer and fail more often than collocated projects. One of the reasons is the increased need for communication, combined with limited possibilities for effective communication.

Software projects developing novel products or new customer specific software are often faced with uncertainties regarding, e.g., both requirements
and implementation technologies. Remote sites or partners need to be involved long before these uncertainties can be resolved. In such projects, the parties cannot receive clear requirement specifications at the beginning. Instead, close cooperation and communication between the parties is required during the whole project, as the project both builds software and tries to understand what to build at the same time. In these kinds of projects, problems often arise, since practices and processes needed for collaborating across distances and organizations are neither well understood in theory, nor typically established in practice.

In our experience, companies easily underestimate the need for specific collaboration practices and processes when running projects across distances. Instead, companies seem to jumpstart globally distributed projects without first planning how to work together with their remote partners. This can lead to quite problematic situations. Collaborating sites and partners often have differing cultures, processes and practices that may pose significant challenges to joint projects.

During the MaPIT research project we have empirically collected experiences both from challenged and successful distributed projects. In this book we report those analyzed experiences. We hope that these collected practices and pieces of advice will help managers and teams better plan and execute globally distributed software development projects.

Intra- or Inter-Organizational Collaboration?

The term global software development (GSD) includes both intra- and inter-organizational collaboration between the countries involved. Inter-organizational collaboration across company borders often involves even more challenges than collaboration inside a single company, e.g. related to information security issues and contracting. In the MaPIT research project we focussed on internal company projects, since most of our partner companies had mainly such projects. However, we believe that the experiences and advice presented in this book can be applied to both kinds of projects, since collaboration and communication are needed, no matter whether the collaboration partners belong to the same or different companies.

Most of our case projects had two or three collaborating sites with the main site located in a Scandinavian country and at least one site in a country with lower wages, such as an Eastern European or an Asian country. In this book we call the main site on-site and the other site or sites often located in lower wage countries off-sites. In our case projects, the on-site held project ownership and was mainly responsible for the customer contacts (since the main customer or customers were often located in the on-site country), and
most domain and system experts were located there. The off-sites provided our case projects mainly with development and testing resources. Both on-sites and off-sites had resources working concurrently for the studied case projects, thus active collaboration between these sites was needed in all the projects.

### Changing Directions of GSD

Our research group has been involved in GSD research since year 2000. During these ten years, GSD has become increasingly more common among software development companies. Thus, we think that distributed projects are here to stay. However, the benefits expected of GSD will not be realized, if companies do not learn how to manage this challenging type of projects well. In particular the planning of collaboration practices between project participants is essential for success. Lately, we have noticed that many companies have experienced managers who have learned, by trial and error, how to manage these kinds of projects and what kinds of practices really work. Unfortunately, these experiences have still not been transformed into common knowledge among all project managers and project team members. During the last ten years we have also observed changes in the ways Finnish companies manage globally distributed software development projects. Next, we will discuss some of these findings.

**Short iterations.** Iterative processes seem to be replacing more traditional, waterfall type process models in distributed projects. The iteration lengths are also becoming shorter. Some years ago, moving to iterations between 1-3 months was already a major improvement compared to the traditional model. Nowadays, commonly used iterations in distributed projects are between 1 and 4 weeks.

**Agile methods.** The latest trend is to apply agile methods, especially Scrum, to distributed projects. Agile methods were originally developed for co-located use and are heavily reliant on face-to-face communication. Thus, at first sight it might seem that agile methods and distributed development do not fit together. However, many projects have noticed that modern communication technologies make it feasible to apply agile methods to distributed projects with good results. Agile practices, based on frequent communication, “force” a distributed team to communicate a lot, which seems to be a major contribution that agile methods bring to distributed projects. In Chapter 3 we discuss how four distributed projects successfully applied Scrum.

**Frequent collaboration and communication.** Earlier, companies often tried to use a black-box type approach for working with remote partners. They handed over a big task and waited for the results, sometimes for sev-
eral months, after which the results were often a disappointment: the requirements had not been understood correctly and the quality was not what was expected. Nowadays, frequent collaboration and communication is expected, and remote partners are brought into the project already during the early planning phase.

Cheap resources or expertise? Resources hired from off-site locations, often situated in lower wage countries, start to have more skills and experience than earlier and have high ambitions. Thus, they are not only low paid, low skilled coding and testing resources but can, with time, be given demanding tasks. As their ambitions grow, off-site personnel expect to get more demanding tasks and fast career development. If this is not possible within a project or company, these employees will be eager to leave for better opportunities. All persons thus lost mean extra training costs for the project and the company.

Rising wages. Wages are quickly getting higher in "traditional" low cost countries such as India. A global trend is to start moving GSD projects to other countries that still have low wages and developing education systems. However, this movement has its’ costs and the wages in these ”new” countries are going to get higher at a fast pace. After some years, there are not going to be any more countries where to move. Thus, low wages will not be enough to survive in the global competition. There is a clear need to make GSD projects much more efficient to keep up with the competition.

New division of work? In higher wage countries there is a fear of losing jobs. Based on our current study, so far these fears have been largely unnecessary, since distributed projects require a lot of work to be done at the high wage on-site countries. That work includes planning, design and lots of communication both with external customers and the remote sites. However, in the future, companies might need to deal with these issues. For example: How is work going to be divided between high and low wage countries, at on-site or off-site locations? Or: Who is going to take care of communication and contacts with external customers?

Improved communication tools and connections. Communication and collaboration tools are getting better all the time, and connections to distant sites are improving, as well. Thus, arranging distributed meetings with tele-, web- and videoconferencing is proving to be a good alternative to travelling and an efficient way of working. Earlier, even voice connections could be of low quality, and if compatible videoconferencing equipment were available, the low quality picture was almost useless. Nowadays, arranging an ad-hoc meeting from one’s own laptop with web cameras is already possible for many, and the set-up time is minimal. With new web- and videoconferencing solutions it is really possible to see the faces of participants and their facial
expressions, which brings a feeling of presence to the meetings.

What have we learned during the MaPIT project?

In this book we present the most important lessons we have learned about GSD during the MaPIT research project. Next, we summarize some of the most important issues found. While these in principle are very simple, recognizing them and implementing them in practice seems to be difficult.

Sufficient planning. A distributed project needs more planning than a traditional, collocated project. In particular, it is important for a distributed team to plan together how to work and communicate. We noticed that this is often neglected, leading to problems later on.

Time and resources. Collaborating and communicating with colleagues requires much more time and resources if a project is distributed than in a co-located situation. This seems to come as a surprise to many projects. The overhead related to communication and learning to work together need to be taken into account when resourcing a project. A distributed project also always requires face-to-face meetings, thus the travelling budget should be planned accordingly.

Simple practices. The communication and collaboration practices that seem to work the best are quite simple, and the needed tools are already available in most companies. However, these practices seem not yet to be common knowledge among project teams. Quite often the same mistakes are repeated over and over again. The project managers having experience from several distributed projects have learned what kinds of practices really work in different situations by the trial and error method. Why isn’t this common knowledge?

Ramp-up activities. Jump-starting a new distributed project is a serious mistake. Sufficiently planned ramp-up activities, including, e.g., planning of collaboration practices, hands-on training, knowledge transfer, collocated working periods, and kick-off meeting, are a better start for a distributed project.

One team? In many of the projects we noticed a mental distinction between the sites as ”us” and ”them”, which often brought a negative atmosphere to the project. This kind of division easily appeared between the sites from low and high cost countries, as one of the bitter comments from our interviews show: ”Finland the brain – offsite the worker”. Creating a feeling of a single team with common goals is important. However, this is not an easy task, but definitely easier in the beginning of the project than later on when facing problems. Face-to-face meetings and collocated working periods are suitable practices that help on the path towards this goal.
Group dynamics. People are the most important asset in a software development project – they lead the project to success or failure. It is essential that team members are capable of working together and that the team’s group dynamics work. Problems in group dynamics show up, e.g., as problems in communication, low motivation, or low quality work. They are easily but falsely explained by other reasons, such as cultural differences. The real problems might be difficult to see and good solutions hard to find.

How to read this book?

This book consists of separate chapters written by various members of the MaPIT research project, according to their specialty. The chapters have been grouped into three parts: Planning and Practices, Teams and Individuals, and Tools and Infrastructure. Each part or chapter can be read as such, in any order that interests the reader. Most of the chapters have boxes or tables summarizing the most important contents of the chapter, thus the busy reader can skim the main findings by just reading these.

In Part I, ”Planning and Practices”, we concentrate on practices related to collaboration, communication and project ramp-up activities. In Part II, ”Teams and Individuals”, we discuss the problems and solutions behind group dynamics, such as trust and distrust. Finally, in Part III, ”Tools and Infrastructure”, we present findings and theories behind the choice of collaboration tools.

The book has been written with practitioners in mind. Thus, for readability reasons, referencing is scarce. Moreover, the research methods employed are not discussed in the chapters, but are described in the Appendix.

We hope that you enjoy reading the book, and that you find something that can help you succeed in global software engineering. We welcome any feedback you might have – please send it to sprg@soberit.hut.fi.

Espoo, 5.11.2010

Maria Paasivaara and Casper Lassenius
Acknowledgements

This book summarizes the lessons learned during the MaPIT (Management, Processes and IT Support for Globally Distributed Software Development) research project of the Software Business and Engineering Institute at Aalto University School of Science and Technology. During three years (1st January 2007 – 31st March 2010) the MaPIT project studied how globally distributed software development projects are managed, what the most common problems are, and what kind of solutions and practices could help projects to succeed.

The MaPIT project was funded by Tekes (the Finnish Funding Agency for Technology and Innovation) and the participating five companies. In addition to funding, our industrial partners provided us with a possibility to study their distributed software development projects, exchanged their insights and experiences, and commented on our findings. Especially, we would like to thank the MaPIT steering group members for participating in discussions and identifying many projects to study. Our warmest thanks go to all persons who worked or still work for the projects we studied, and who were able to give some of their valuable time for the interviews, as well as all those persons who participated in our feedback sessions and experience exchange workshops. Finally, we would like to thank our international partners, Dr. Peter Gloor from MIT and Dr. Roberto Evaristo from 3M for participating in our project and providing us inspiration and help in many different ways.

With the very best regards from the MaPIT project team,

Maria Paasivaara
Tuomas Niinimäki
Arttu Piri
Peitsa Hynninen
Nico Hiort af Ornäs
Casper Lassenius
Shosta Sulonen
Part I
Planning and Practices
Chapter 1

Initiating Global Software Development

Nico Hiort af Ornäs, Casper Lassenius and Maria Paasivaara

1.1 Introduction

This chapter discusses how to set up a new collaboration in global software engineering, and how to successfully start a project. The chapter is divided into two main parts: setting up the overall collaboration, and initiating a project.

1.2 Setting up the collaboration

When starting collaboration with a new organization or team in global software engineering, there are several things that need to be dealt with in order for the undertaking to have a good chance of success. These include clearly communicating the overall strategy and reasoning behind the collaboration, as well as building trust between the main site (on-site) and the remote site or sites (off-sites).

1.2.1 Communicate the overall GSD strategy

When starting a new global software development relationship, it is crucial to clearly and openly communicate the overall strategic intent and plans, as well as deal with any concerns raised by personnel, both at on-site and off-site.
CHAPTER 1. INITIATING GLOBAL SOFTWARE DEVELOPMENT

Deal with fear

While there are many good reasons for transitioning to global software engineering, cost-savings — either real or imagined — are often the main driver. This can create negative feelings such as fear and distrust that can severely hamper collaboration if not dealt with in a proper way.

At the on-site location, people often fear losing their jobs, as they think they will be replaced by cheaper workers from abroad. If this fear is not dealt with, it can lead to negative behaviors aiming at undermining the distribution so as to save the jobs of people at on-site. In practice, such behaviors include a wide range of actions and non-actions, from passivity in communication to knowledge hiding and direct sabotage. In effect, people at on-site will make sure the distribution fails if they are not at least to some degree convinced that it is safe for them to participate.

At the off-site location, people might consider themselves less worthy or important, as they may think they are involved just because they are ”cheap resources”. This in turn, might lead them to be poorly committed to the collaboration, and to have low motivation for performing high-quality work. In some cases, on-site personnel might even treat their off-site colleagues as ”second class citizens” which is very demotivating. In our experience, these reasons, as well as the lack of sufficient information to do a high-quality job, more often explains the perceived low quality of work at off-site than actual incompetence. However, in our studies, on-site personnel often complained about the perceived incompetence of off-site teams, despite the fact that they had not transferred necessary information to the off-site team, which might have actually resulted in the perceived incompetence.

The experiences in the projects we studied in the MaPIT research project show that it is very uncommon for people at on-site to actually lose their jobs to off-site. Building and sustaining a working collaboration between several sites adds an often surprising amount of overhead related to communication and coordination.

Take a long-term view

”Going global” is not an easy quick-fix to domestic cost problems. It is important to take a long-term view of global software engineering, and to understand that it takes both time and effort to build a sustainable, working collaboration between sites in different countries. During the startup period, which can range from a few months to a few years, productivity is likely to be lower, and costs might be even higher than when working in a non-distributed way.

Companies without experience tend to be overly optimistic and often
CHAPTER 1. INITIATING GLOBAL SOFTWARE DEVELOPMENT

quite unrealistic in their expectations of immediate and large cost savings and improved product quality when initiating their first global software engineering efforts. Understanding that making global software engineering work requires skill, investment and patience is crucial to success.

A long-term view also means that you need to work out how to, in the long term, divide work between your home country and the off-site locations, so as to derive maximum value of the collaboration. Likewise, you need to think not only about motivating and creating sustainable career prospects for your personnel at your home site, but you should also be concerned about how to motivate the off-site personnel, so as to keep them committed to working for you. This is important, since in all but the simplest projects, knowledge transfer to off-site personnel, is a crucial, difficult and costly activity. Every person who leaves requires a new person to be trained as his/her replacement; this is no less expensive or difficult just because the people are in another country. Thus, it is important to care for the off-site personnel as if they were local.

Developing and communicating a positive long-term view can make a huge difference in making a collaboration just barely work or being hugely productive. This issue is touched upon in many of the following sections, in which we discuss mechanisms such as gradually giving more responsibility to off-site and dividing work in a collaborative manner. Thus we also demonstrate our view that to be really successful, global software engineering must be viewed as a collaborative effort, in contrast to a pure client-subcontractor model.

1.2.2 Division of work

Build one single project team

The overall goal when designing working practices for a distributed project should be to make everybody on the project feel as if they belonged to a single project team, even if the project were so big as to require subteams. In particular, one should try to avoid having a setup in which there is the on-site expert team that delegates simple tasks to the ”less capable” people at off-site. Achieving team integration in the level where everyone on the project feels equally important is difficult, but most of the practices discussed in this book, from co-located sessions to frequent communication and the use of cross-site teams help facilitate this overarching goal.
Consider using cross-site teams rather than single-site teams

A major decision that needs to be made with respect to organizing a global software engineering project is to use site-specific or cross-site teams. Site-specific team members are all from a single site, whereas cross-site teams are composed of members from different sites.

When every site has sufficient information to work on, e.g. different modules of a product, and the need for inter-site communication is limited, the most efficient solution is site-specific teams.

However, according to our case studies, on-site people often possess deep knowledge related both to technical implementation and domain that is needed for the development. In such cases, the use of cross-site teams can be a superior solution, since active collaboration and communication between the sites is needed. In addition, having cross-site teams helps avoid the ”us” versus ”them” mentality between the sites and forces people to communicate frequently with their off-site colleagues in order to be able to work effectively.

Consider starting with system testing

When outsourcing an existing product for offshore development, it is best to have the off-site personnel test the product first: that will help them get more familiar with the product as well as gain business domain knowledge and more understanding about what the system is supposed to do. If the product is complex, and testing thus requires deep domain knowledge, or if the user-interface is in the native language of the on-site that the off-site personnel cannot understand, this might of course not work in practice without adequate support.

Have testing and development at the same site

As the off-site team gets up to speed, having testing and development at the same site is preferable. This has the benefit of increasing collaboration between testers and developers. Another benefit is that there will be less effort needed to transfer a testable product from development to testing. Implementing this does, however, require careful planning regarding the division of work. A two-tiered approach is usually used: the coders test their own code, preferably using automated tests, before handing it over to testers for higher level testing.

Some amount of collaboration is usually needed between testers and developers. A couple of the case projects we studied had separate testing teams at the on-site location while the development teams were at the off-site lo-
CHAPTER 1. INITIATING GLOBAL SOFTWARE DEVELOPMENT

cation. This generated several problems. First of all, the testing teams at on-site reported that most of their time was spent documenting and reporting trivial bugs. These trivial bugs could have been easily revealed at the off-site location had there been at least some degree of testing performed locally. Secondly, when testing and development are separated, testing of the product will usually not begin before the work has been handed over from the development team to the testing team. However, in many cases it is preferable to start testing already during development. This is easier if the people doing development are co-located with the testers.

Separate maintenance work from product development work

By having a separate maintenance team, the team can quickly react to support requests and thus be able to give a rapid response to customer bug reports or change requests. This allows the rest of the development team to solely focus on developing the main branch. If the support requests require much collaboration with external parties, such as customers, then it might be better to locate the support team at the on-site location. One example of this is work that requires customer-specific configuration or development, or that must be done in the customer’s native language.

In one of our case projects, a separate maintenance team was located at on-site. This maintenance team had a development cycle different from those of the other teams. The development cycle of this maintenance team was a short two-week cycle, allowing them to do bi-weekly releases. The off-site location had only development teams that were developing the main branch of the product. As such, the amount of collaboration needed between the sites decreased as frequent change requests did no longer require work from the off-site team. Furthermore, the other development teams were given peace-of-mind, since they could completely focus on new feature development.

Define clear roles and responsibilities

A clear organization structure in any software development project allows the project members to better understand everybody’s responsibilities. This is even more so in a distributed environment, as the transparency between sites is always limited. In order to efficiently plan the division of work and responsibilities, there should be a clear organizational structure that is clearly communicated to everyone involved in the project. This can be done, e.g., with the help of a project website, the use of social media, as well as by encouraging and supporting face-to-face meetings.

One of the case projects completely lacked structure and hierarchy. The project was organized into two teams, on-site and off-site. In this type of a
project, most of the communication between the sites was channeled through the project managers of each site. The off-site developers were unsure about who, from the on-site team, had designed the analysis models and written the specifications. This hampered the communication between the sites and made the off-site developers hesitant to ask questions from on-site.

1.3 Starting a GSD Project

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. As such, it is better to first solve the problems related to project management than to introduce new challenges related to product development. Next, we will first discuss arranging a project kick-off, followed by thoughts on knowledge transfer and the set-up of working practices.

1.3.1 Arrange a project kick-off

At the beginning of a project, it is often a good idea to arrange a project kick-off event or meeting, preferably face-to-face and involving as many of the project team as possible. A face-to-face kick-off gives the project team members a chance to meet each other before starting the collaboration. When later communicating as a distributed team, most people will feel awkward if they have not met before. Thus, forming basic social ties is one of the major goals for a project kick-off. If the project is big, it might be unfeasible to arrange for everybody to meet face-to-face in a single kick-off. In such cases, smaller face-to-face events, e.g. one for each sub-team, can be arranged.

During the kick-off, the project background, goals and technology are typically presented in order to ensure a joint understanding of what is to be done and why. Integrating knowledge transfer activities, discussed in the next section, with the kick-off is frequently useful. For example, having one day of presentations of the project goals etc. might be followed by several days or even weeks of co-located knowledge transfer activities, including both training sessions, as well as learning-by-doing. Having people from different sites working together in a co-located fashion also facilitates socialization and forming of good communication habits as well as helps building trust. Arranging only one-day kick-off meeting with all travelling involved
might not be cost-efficient. Thus, proper planning and combining of start-up activities are needed.

In one of our case projects, there was no joint kick-off between the on-site and off-site teams; only the top management from each site had a meeting in the beginning of the project. Several of the off-site project members complained that they were never given a proper introduction to the project, e.g. what the project is about, what the goals are, what will be the next steps, etc. Instead, they were told only to read some guidelines on how to work on the project. This had severe implications on their motivation to work in the project, and the lack of personal ties prohibited the sites of communicating efficiently with each other.

1.3.2 Invest in knowledge transfer

In cases where a new off-site is brought on, transferring knowledge from the on-site team to the off-site team is one of the most important activities that should take place in the very beginning of the project. Knowledge transfer at the beginning of the project will allow off-site to gain knowledge about the product and business domain. Knowledge transfer activities should also convey information on working and communication practices, as well as on how to use development and testing environments and frameworks.

"I think this [kick-off training] should be the first step when someone starts working on a project. And I think this is happening in other projects at [off-site location] . . . when someone comes to a new project or to some on-going project, he’ll first go to [one of the on-site locations] for at least a month."

– Off-site developer

Although the early phases of a global software engineering project will have a special focus on knowledge transfer, it should be understood as an ongoing activity. While it might seem logical to do most, if not all, knowledge transfer right up front, one must remember that, for any even moderately complex product, it is unlikely that a short up-front training can convey all the necessary information to get new people up to speed.

Initial knowledge transfer is preferably undertaken in co-located hands-on training sessions combining theoretical lecture-style teaching with a hands-on learning-by-doing approach. Working hands-on, preferably on real but simple development tasks, is more motivating (and productive) for the people learning the system than purely theoretical teaching would be. Doing knowledge transfer face-to-face, possibly involving parts of both the on-site
and off-site teams, provides the project team with possibilities to socialize and engage in team building activities, as well. Allowing the project members to build trust and personal relationships with each other will be very beneficial, as it will enable more active communication between the sites once team members are distributed.

In one of the case projects, a four-week long knowledge transfer session was held at the on-site location. The sessions were both theoretical and practical, allowing personnel from the off-site location to first get the big picture about how the system works, and then go into more details such as how to code the system. As the knowledge transfer period lasted for a total of four weeks, the project members had enough time to socialize and build personal relationships with each other. The knowledge transfer period also included social activities such as having dinner and going to the sauna. Some of the project members saw the team building activities as more important than the training sessions, because these established relationships helped enable efficient collaboration between the two sites in the later stages of the project.

Arranging this kind of knowledge transfer periods can be done at any site, depending on the situation. In order to make the physical co-location as efficient as possible, it can be a good idea to distribute training material beforehand. In complex projects with lots of need for knowledge transfer, periods of training and hands-on learning on the job should alternate. One should be careful not to include too much information in short training periods, as information overload can hamper learning.

Knowledge transfer sessions should not only be held in the beginning of the project, but also on-demand. One case project held a handover workshop 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 on-site 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 later on.

A contrasting example can be found from another case project that did not organize any knowledge transfer in the beginning. Instead, the off-site team members only received a specification document that they could study by themselves. The project members from off-site complained several times about not having enough knowledge of the business and the product to be able to complete their tasks properly.
1.3.3 Increase off-site responsibilities gradually

In order to keep the off-site personnel motivated, and to maximize cost savings, gradually moving more and more responsibilities to off-site locations can be a good overall strategy.

The tasks, activities and responsibilities that are done at off-site should be increased in a step-by-step fashion. The first task for the off-site personnel should preferably be simple, e.g. making small and easy changes to the software product or to perform testing. This will allow off-site to demonstrate that they are capable of working and delivering results. 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 or to find knowledge gaps that must be bridged for the project to continue successfully.

Another benefit of this approach is that the off-site team is able to set up their working practices at the same time as they are working on their first tasks. Indeed, many distributed projects fail because they do not invest enough time in a successful project setup.

After the off-site personnel has demonstrated a capability of working in the project and the project team has set up their working practice, 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 those 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 creating a chaotic environment.

In one of our case projects, an impressive amount of work was gradually transferred from on-site to off-site. In the end this project managed to have around 80% of all the activities at the off-site location, whereas this percentage in the other projects was somewhere around 40-50%. The first responsibility moved from on-site to off-site was testing. Eventually, this project continued to move other responsibilities such as product architecture, requirements management and acceptance testing. This was possible because the on-site team had become aware of the amount of skills and competence available at the off-site location. However, an approach like this takes time and patience. In this particular case project, it took over 18 months for the on-site team to transfer the responsibilities to the off-site team. In the end, the number of personnel at the on-site team was five, while at the off-site team it was 25. According to one off-site project manager, increasing the responsibilities in steps helps keeping the off-site team members motivated:
"Start with body shopping, but move to service orientation with time. Okay, in the start probably it is better to assign a 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

In addition, this project was the only one we studied in which off-site personnel were directly involved in customer contacts, which was another major motivating factor for the off-site personnel. In the beginning of the project, all customer communication was channeled through the on-site team. Later, the off-site team was allowed to participate in the meetings between the on-site team and the customer: first, the off-site team took a more passive role, just listening in and getting information directly from the customer, but eventually it was allowed to communicate directly with the customer without the involvement of the on-site team. However, establishing this kind of relationship requires time. In the end, moving as much of the work as possible from on-site to off-site might introduce 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."

– Project owner, on-site

1.3.4 Involve the whole team in planning

In order to get people committed, it is often a good idea to hold planning sessions in a group consisting of project members from all sites. In these sessions, tasks can be created and divided among the team members. It is important to communicate any task priorities to the developers and ask them for their input regarding the work estimates. This helps commit the developers to the tasks and their estimates as well as give the project manager a chance to reprioritize the tasks in case the estimates change drastically.

We noticed one hazardous example of how not to transfer work between sites. In this project, the allocation of work was done mainly through a task management tool. In the beginning of the iteration, the on-site team would create tasks, put them in the task management tool, and then send the specifications to off-site via email. The opportunity for the off-site developers to
ask any questions regarding the tasks was close to nonexistent. In addition, the tasks did not have any priorities whatsoever. Because of this, the off-site team spent a lot of time on issues that were considered unimportant by the customer. Off-site’s involvement in any iteration planning activities was nonexistent also.

In another case project, we noticed that including the off-site developers in the iteration planning sessions increased their motivation to complete their tasks, as the following comment from an off-site developer shows:

"In this project, we have been usually planning iterations together with the leaders, sometimes even without any leaders. Right now when we are a small team (...) so we are planning it all together, with on-site as well (...) so really having a one-hour phone conference with Microsoft Project and Word-document, use-case and really planning it together - so, guys are, I think, more committed because they are setting deadlines for the tasks, they are setting the effort that they will need for a specific task. They are saying yes, I will complete this task in eight hours."

– Off-site developer

1.3.5 Implement good working practices and processes

Distributed projects tend to be of a reasonable size. Thus, it is important to understand that the processes and working practices cannot be ad-hoc, as they often are in smaller co-located projects. Establishing working practices that will satisfy the needs of both sites requires planning, effective communication, and often also time and patience. These practices should be discussed with the whole or part of the team so that both sites would commit themselves to them. After all, it is in the interest of all the sites to establish working practices that allows everybody to work efficiently. Below is a comment from an off-site project manager:

"So several times there wasn’t enough communication to understand that, for example, processes are a must in a distributed environment, so that it cannot be so ad-hoc. Otherwise we will fail with big problems."

– Off-site project manager

A few of our case projects used the Rational Unified Process (RUP). As with most software processes, RUP is not a concrete process that automati-
cally fits the requirements of every project, but rather an adaptable process that requires tailoring to the specific situation in a project. This had been realized in some of the case projects, while in others not. Moreover, most software processes have not been designed to work in distributed environments. Special care should be taken to ensure the needs of all sites when tailoring the processes.

1.3.6 Designate liaisons

A liaison is a person that moves from one of the sites to another for a long-term period. Doing this can introduce several benefits, such as improved communication and higher transparency between sites. The liaison brings his or her knowledge of the people, culture and working habits of his or her site to the other site, and can therefore greatly help improve understanding, trust and communication between the sites.

In one case project, an off-site team member was moved to the on-site team. This person had several contacts at the off-site location and knew most of the people there. Thus, whenever someone from off-site had a problem and did not know whom to contact at the on-site location, he would instead contact this liaison. The liaison was then able to work as a proxy between the sites as he knew most of the people from both sites.

Another benefit of this liaison was that he was also able to work as a proxy 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 sometimes were even afraid of speaking in English. This cultural liaison was then able to work as an on-demand translator between the team members during the meetings.

"I think that during training they found that it might be useful for the project if they had someone here [from off-site] who knows the situation there [at off-site]. So one of my tasks is also to serve as a proxy between [off-site] and [on-site]."

– Cultural liaison

The third benefit was that the work at the on-site was becoming more transparent to the off-site team. The liaison was able to communicate to the off-site the true feeling of how the project was progressing at the on-site:
"One time it looked like that [the customer] is going to cancel this project, and [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 assuring him [off-site project manager] that now "Hey, it is OK already, so you can trust that it’s going to continue."

– Cultural liaison
CHAPTER 1. INITIATING GLOBAL SOFTWARE DEVELOPMENT
Chapter 2

Communication and Collaboration Practices

Maria Paasivaara and Casper Lassenius

2.1 Introduction

This chapter concentrates on communication and collaboration practices in GSD projects. Communication is often seen as the biggest problem in distributed development, since e.g., geographical distance limits face-to-face communication and time-zone differences prohibit asynchronous communication. First, we briefly discuss communication challenges that distributed projects typically have. Then, we present communication and collaboration practices that the case projects we studied used successfully.

2.2 Challenges

Lack of informal communication. In collocated projects informal communication, e.g., when drinking coffee or having lunch, distributes information about the project state, problems etc. However, in distributed projects all informing requires extra effort. It is easy to forget distant colleagues and forget to tell them, e.g., about changes. When they are out of sight, they are also easily out of mind. Misunderstandings. Misunderstandings are common in distributed projects, partly because of language and cultural differences, and partly because of lack of communication. For example, specifications can be easily misinterpreted, and on-site personnel might not understand to provide off-site personnel enough context information to make them able to understand other messages in that context.
CHAPTER 2. COMMUNICATION AND COLLABORATION . . .

**Cultural problems.** Cultural problems arise especially in projects where persons from different countries are participating. However, also corporate cultures can be different, causing problems. Cultural problems are closely related to communication and can cause misunderstandings and lack of trust.

**Limited travelling.** In distributed projects traveling is always needed, especially in the beginning of the project, to get to know each other, and e.g., in integration and problem solving situations. Traveling consumes both time and money, and thus incurs extra costs and delays to the project. Sometimes managers do not understand the need for face-to-face meetings and believe that they save money by restricting travelling. However, we noticed that our case projects that had a reasonable travelling schedule did not have as many problems as the projects where travelling was almost non-existent.

**Lack of trust.** Trust is needed between collaborating partners to be able to communicate and work effectively. If trust is lacking, the project may face serious problems, e.g., all needed information might not be communicated, or partners might not be helped, when trust is lacking. Building trust takes time and face-to-face communication, which are rare in distributed projects but provide the best opportunities for trust building.

**Problem solving takes time.** Solving problems is easiest in face-to-face meetings. For example, using email for problem solving can take a long time if several clarifying questions need to be asked. In distributed projects problem solving can also be delayed because developers do not know whom to contact and then try to solve the problem on their own. Especially off-site developers, if they do not have enough context information, may finally end up with altogether wrong solutions when trying to solve a problem on their own.

**Differing processes and working practices.** These differences can cause problems when partners forget to discuss and agree about them, expecting that everyone else has similar processes and working practices than they themselves have. Since communication does not occur naturally in a distributed project, also communication practices need to be planned across sites.

**Differing tools and communication technology.** Often it is essential to solve the differences regarding tools, otherwise working with distributed partners is difficult. All collaborating sites need to use interoperable communication tools, such as chat or videoconferencing. Also their development tools, as well as version and configuration management tools, need to be compatible. **Time-zone differences.** Communication suffers a lot if time-zone differences between the sites are large. If the daily working hours between the sites do not overlap or the overlap is very brief, it is hard to arrange synchronous communication, such as tele- or videoconference meet-
ings, between them. Moreover, problem solving takes even more time: e.g., solving a problem by sending emails can take days, while when using phone the same problem might be solved in a few minutes.

2.3 Practices

Next, we turn our attention to practices that can help solve the communication problems discussed above. These practices include, e.g., frequent communication, increasing team responsibilities, and use of iterative development. Many of these practices can be found in agile software development, which seems to be a good fit with distributed development if adopted appropriately. However, many of the practices discussed can be used even if you do not apply them to agile development. Several of our case projects were moving towards agile work methods, and four of the studied projects successfully used an agile method called Scrum. In Chapter 3 we describe in more detail how to apply Scrum practices to distributed projects.

The communication and collaboration practices described next are suggestions collected from our case projects. The aim of this list is to provide ideas of possible practices that can be used in your own projects. However, each project is different, thus how to apply the practices to a project in question need to be discussed and designed case by case. Moreover, this list is not complete: there might exist many more good practices. Also, a single project does not need all these practices presented. A suitable set of practices needs to be chosen and applied to the project in question.

2.3.1 Use frequent deliveries

Incremental integration and frequent deliveries is a core practice in agile methodologies for collocated projects and can also be applied to distributed development. Indeed, distributed development can gain many benefits from using an iterative model, e.g., by providing increased visibility into project status. Our case projects used iteration lengths between 2 and 4 weeks. We observed several projects that had started with a more traditional process model but, due to problems, had moved to iterative development. Many of the projects also used continuous integration and automated testing.

There are several benefits in using an iterative process with continuous integration and frequent deliveries:

- **Transparency of progress.** Frequent delivery cycles and integration bring transparency of work progress to all partners. In the projects where both the on-site and off-site used iterative development, the
off-site regularly delivered functioning code during the development phase. Our interviewees told that when deliveries were integrated and tested right away, this gave everybody a very good picture of how the project was progressing. They had noted that frequent deliveries made it easier to monitor the real progress, which otherwise would have been more difficult, especially regarding the off-site.

- **Instant feedback.** Integration and testing reports give developers instant feedback on their work and is a good motivator. This seemed to be important especially for off-site developers, who otherwise seldom got feedback from the on-site. Moreover, when the on-site personnel sees that the people at the off-site are doing high quality work, the on-site people start to trust and respect their colleagues at the off-site locations, which makes further collaboration easier.

- **Flexibility.** From the on-site’s point of view, this way of development brings additional flexibility, as changes can be made during the development phase without time consuming negotiations and cumbersome change management procedures. Of course, a suitable type of contracting has to be chosen between the sites or partners. Iterative development also makes it possible to involve the off-site into the project in the early phases of development when requirements cannot yet be specified in detail. Since requirements are allowed to change during the project, work can start despite technological or goal-related uncertainties. However, this requires all parties to have ”an experimental mindset” and practice efficient and open communication.

- **Ensuring understanding of requirements.** Frequent integration and testing can help ensure that the off-site personnel have understood the requirements correctly. This is a typical uncertainty in distributed development, especially when the sites have not worked together before and have different cultures and the off-site personnel has no deep understanding of the context and end-customer’s business. Frequent integration and testing provides fast feedback, any misinterpretations become visible early, and possible misunderstandings have less damaging consequences. Moreover, learning from mistakes is fast and happens early, thus preventing problems from accumulating and creating situations that can be difficult to resolve.

### 2.3.2 Reduce temporal distance

Temporal distance can cause major problems by restricting possibilities for synchronous communication. For global software development to work well,
frequent and preferably daily synchronous communication between sites is needed. The larger the time-zone differences between the sites, the more difficult it becomes to arrange this.

The most basic approach to dealing with temporal distance is to select partners from countries in the same time-zone or not more than a few hours apart, so as to maximize the common working hours. Even if the time-zone differences are small, the possibilities for synchronous communication are affected, e.g., by lunch hours and cultural issues such as the times the working days start and end in different countries. Thus, e.g., a two-hour time-zone difference between the sites can provide only two to three hours real synchronous working time.

If forced to work over large temporal distances, some companies have chosen to alter the working hours of their personnel to create overlap. For example, many Indian software companies working with the US ask their personnel to work according to the US time zone, thus creating good opportunities for synchronous communication. In one of our case projects, the interviewed personnel from a Russian subcontractor were very happy to work with a Finnish company because the company’s location allowed them to work during daytime, whereas many of their colleagues in the same company working for US customers had to put up with working during night on their customers’ request.

Some other companies may choose to have personnel at both ends slightly alter their working hours to provide possibilities for synchronous communication.

2.3.3 Reduce the number of sites

As the number of collaborating sites increases, the communication overhead, as well as communication challenges increase. The optimal number of sites is one, of course, and two is better than three. Some of our interviewees stated that they really try to limit the number of collaborating sites to a maximum of three, since four or more sites make the communication and coordination overhead so large that the project will most probably face huge challenges and work will become inefficient. In most of our case projects the number of the collaborating sites was just two.

2.3.4 Reduce cultural distance

In addition to temporal distance, cultural distance, both with respect to national culture as well as corporate culture, needs to be taken into account. Here, again, the general advice is to try to minimize cultural differences.
Our case studies show, e.g., that Finnish companies have a much easier time working with companies in other European countries than, e.g., with Indian companies. This seems to be mainly related to cultural issues. While it is not always practical to choose a partner from a country that is culturally close, reduction of cultural distance should weigh in on the decision if at all possible.

Practical means that can be used to help reduce cultural distance include the use of cultural liaisons, bridgehead arrangements, internalizing foreign entities, language and cultural training, and rotating managers. Cultural liaisons are persons who travel often between key projects sites or move to work for other site for a longer time period. They might be, e.g., managers who during their trips facilitate communication and mediate conflicts between sites or experts that stay a longer time at the other location and facilitate collaboration between the sites. A bridgehead arrangement is a suitable alternative for companies working with offshore centers. It means moving part of the off-site personnel to work at the on-site for a longer time period, even for the duration of the whole project. For example, 25% of the off-site personnel might be situated at the on-site close to the customers to take care of the communication with them, while the rest would work at the off-site. None of our case companies used the bridgehead approach. However, many of our case companies used a cultural liaison, which seemed to be a very successful and sufficient practice for them, since in most of the cases local on-site personnel took care of the communication with the external customer. Internalizing foreign entities is a solution to organizational culture problems, suggesting that instead of using subcontractors the company could build or buy their own foreign entities, thus helping to reduce corporate cultural problems. This was actually the case in most of the case projects in our study. Most of our companies had established their own off-site companies. Only in two case projects the main off-site partner was a subcontractor company. Arranging language and cultural training for staff is one solution to language and culture related communication problems. Finally, a few of our case companies had used the rotation management across locations, especially from on-site to off-site. This increases the understanding of different cultures and the managers involved can work also as cultural liaisons between the sites.

2.3.5 Define and communicate a clear organizational structure

Clear organization structure is important also from the communication point of view. It is important that all personnel both at the on-site and the off-site
know the organization structure of the project, as well as the responsibilities of the personnel. That way it is easier to determine who should talk with whom and about which issues.

In one of our case projects it was even unclear who was the project manager. During our interviews, when asking different persons about whom they saw as the project manager, we got four different answers! That is, four different persons were named by the personnel as “really leading the project”. This shows that the situation in the project was really messy and the project as a whole was highly challenged. Also, the atmosphere in the project was bad.

2.3.6 Prefer dedicated team members

When resourcing a distributed project, aim at having dedicated resources. Distributed development always involves a high level of communication overhead, which reduces time that can be spent on, e.g., development tasks. Thus, people who are only spending a fraction of their time on a distributed project can be of only little value.

Dedicated team members also help guarantee that information flows as well as possible, since all the members are able to participate in meetings. Finally, dedicated human resources have a better chance of socializing with the other project team members, including resources at other sites, helping to create a feeling of “one single team”.

2.3.7 Create one single team

In order to facilitate good collaboration between different sites, one should aim at making the team members feel as if they formed a single project team. In particular, it seems to be important to avoid having a division between “we” and “them” that easily occurs when organizing a project with site-specific teams. In one of our case projects, a developer from the off-site described the feeling as if the project had had two teams – “on-site: the brain, and off-site: the workers”. As can easily be imagined, collaboration between the on-site and off-site in this project did not work very well.

Practices that help build this feeling of a single team include periods of co-location, project kick-off and hands-on training events, as well as strategically more difficult things, such as involving the off-site in customer meetings, and increasing the responsibilities of the off-site teams.
2.3.8 **Arrange face-to-face team building possibilities**

Having met someone face-to-face makes it a lot easier to communicate with that person using various communication tools. Therefore, face-to-face meetings are crucial, especially in the beginning of a project. It is important that distant sites and companies have “faces”. Otherwise, they are easily forgotten and, e.g., their questions might not be regarded as important and urgent to answer. A high level of trust between parties makes collaboration easier. However, trust develops slowly, most likely in face-to-face situations, which are rare in distributed projects. Building a good relationship between on-site and off-site personnel requires that people in both sides are treated as colleagues and experts in their field. In some cases, we have observed that on-site personnel treated off-site personnel as if they had been second-class citizens, which is a bad base for collaboration.

2.3.9 **Provide background information**

Giving sufficient motivation for and background information on a project can be very motivating to off-site personnel. It is important to make sure that everyone on the project understands what is done, and what the business case for it is. When everyone understands the importance of and business case for a project, commitment is typically improved.

Understanding the business case can also make it easier for off-site personnel to appreciate certain decisions and tradeoffs that might be made by people at the on-site location. In some of our case projects the lack of understanding of the business case for the project made the off-site personnel continuously question decisions made at the on-site. The decisions made a lot of sense from a business point of view, although not from a purely technical viewpoint. Thus, we think that a lot of misunderstandings and conflicts in the project could have been avoided had only the business case been clearly communicated to the off-site team members.

Other important background information can include some level of knowledge of the whole system if the development project is only a part of it, as well as, information of a possible end-customer’s business.

2.3.10 **Agree on communication practices and tools**

In order for communication to work smoothly, it is important that the whole distributed team discusses communication practices and agrees to them. Things that should be discussed include:

- **Meeting schedule.** Meetings must be scheduled well in advance. By
far the best strategy is to have regular meetings that take place according to a predefined and strictly followed agenda. Scheduling meetings on a case-by-case basis in a distributed project can be quite difficult and time consuming.

- **Raising issues.** The project should create a culture in which people are not punished but rewarded for bringing up problems or even potential problem issues as soon as they emerge. This is the best way to make sure that problems are dealt with as quickly as possible, with the lowest overall cost to the project.

- **Giving feedback.** Encourage people to give instant feedback, in particular positive – if something has been done well, say so! Create an atmosphere in which everyone is expected to give feedback to everyone else. It is not only managers or on-site personnel who are supposed to give feedback.

- **Chat practice.** Deciding on the use of instant messaging, e.g. how to use the status field, when to keep the tool open, and deciding on a policy for saving and distributing chats with the larger team.

- **Use of the telephone.** When and for what to use phone calls or other voice-media.

- **Email etiquette.** In particular it is important to decide on how fast to respond to emails, e.g., within 4 hours or within 24 hours. Moreover, it is useful to decide how to use the out-of-office functionality

- **Ad-hoc meetings.** Discuss how to arrange ad-hoc meetings, and what tools to use (e.g., web-conferencing or screen sharing).

- **Communication tools.** Agree on common communication tools (e.g. news groups, chat, videoconferencing)

- **Development tools.** Agree on common development tools and their usage. These include configuration and version management tools, use of joint repository or replication, setting up the test environment, possible use of collocated testing periods in a real testing environment, bug reporting, test reports, etc. Ensure that everyone on the team has access to the basic tools.
2.3.11 Agree on progress monitoring

Frequent monitoring of progress in a distributed project is important. Time sheets, used by some of our case projects, are not really useful for this purpose, since hours used might not correspond to real progress in the project. However, time sheets might be needed for other purposes, e.g., billing the external customer.

To form a proper idea of what is being done in the project, daily or weekly meetings as well as backlogs and burndown charts are much better than time sheets. However, there must be a common agreement about the meaning of the word “done”. Otherwise, there will be as many definitions as there are team members. In our case projects we noticed that especially the on-site and off-site personnel’s views easily differed where no agreement existed about the term. For example, “done” could mean that the code had been written, that the developer had also tested the code, that the code had been integrated and tested and the bug fixing done, or that also the peer code review had been done.

For the project manager as well as for all the distributed team members it is important and motivating to see frequently how the project is progressing. Collocated team members might get a view of the project’s progress by overhearing conversations and in informal communication with their colleagues. However, although remote team members do not hear these conversations, they still would appreciate that kind of information, as we noticed in many of the case projects.

2.3.12 Arrange training

In our case projects, the off-site teams were often involved in the projects later than the on-site personnel. Either the on-site personnel had been planning the project and architecture or even building frameworks before taking the off-site into the project, or they had been involved in developing previous versions of the product. Thus, training of the off-site and sometimes also the on-site personnel is needed in most projects. Besides technical knowledge of technologies and techniques used, the new project team members need to get acquainted with the work already done in the project, as well as with the requirements and background information, e.g. regarding the end-customers’ business. In most situations, sending written material is not enough, but face-to-face training with hands-on working periods are needed. In the previous chapter (Chapter 1) we talked about different possibilities of arranging this kind of knowledge transfer training sessions in practice.
2.3.13 Create visibility across sites

It is important to create visibility across all the distributed sites during the whole project, so that both team members and project managers have a good idea of the project all the time. For a project manager to have visibility to other sites is obvious, but the importance of creating similar possibilities for visibility to team members is not always understood. Team members need visibility to be able to accomplish their tasks as well as possible, to understand the project situation and motivation behind decisions, as well as to be highly motivated for working in the project. Seeing a bit bigger picture than just their daily work should help in this.

An example of the lack of visibility is that responsibility about the customer often remains at the on-site, located close to the customer. Thus, off-site team members might never meet or even discuss with the customer. We noticed that in many of our case projects this led to a situation where the off-site was not committed to the goals and schedule as deeply as the on-site, since they just could not see the importance of this. They had not made any promises to the customer. In one case project the situation was improved when the off-site members were invited to teleconference meetings with the end-customer.

Visibility can include many different things:

- Possibility to follow the project progress.
- Hearing the current problems and solutions from other sites.
- Seeing who are present at remote sites and available for conversations (e.g. one’s chat status can tell this).
- Understanding the business case of the project.
- Understanding the end-customer’s business.
- Visibility to end-customers needs and requirements, as well as reactions to deliveries.
- Seeing the future plans for the project, e.g., the product roadmap.
- Seeing the personal development possibilities that the project offers.

2.3.14 Arrange frequent status meetings

Since transparency is a big problem in distributed software development, having frequent status meetings that help keep everyone informed about the project progress is crucial. As a rule of thumb, the meetings should never be less frequent than once a week, but the most benefits seem to come from short daily meetings, as prescribed by the agile methodologies.
Status meetings should involve the whole team, have a strict and clear agenda, and make sure that everyone participates. The “Scrum way” of doing this seems to work well: everyone is required to report what he or she has done since the last meeting, what will be done before the next meeting, and whether there are any problems. Any discussion or problem solving will be done after the meetings in one-on-one or small group sessions. This ensures that the status meetings are kept short – from less than 15 minutes to half an hour at a maximum when having meetings more seldom than daily. This saves costs, and keeps participant frustration at a minimum. It is important that during these status meetings long discussion on topics that are interesting to only a few participants are restricted to minimum and moved after the meeting. The final result of this kind of off-line discussion can be reported afterwards to the whole team.

For these kinds of meetings to work, it is crucial that everyone participates. Many teams even have people who are unable or do not want to participate in meetings held at their own location. They might not see the importance of the meetings or feel that meetings are inefficient. However, this kind of behavior makes the information flow in the project inefficient and lowers team spirit.

2.3.15 Arrange weekly meetings across teams

Whereas team members participate in frequent, preferably daily meetings, inter-team meetings in larger projects can be arranged a bit more seldom. The Scrum agile methodology uses weekly team synchronization meetings called Scrum-of-Scrums. Such meetings can be adopted even if the project does not use the Scrum methodology.

Also, these inter-team meetings should have a clear agenda and be conducted efficiently. The participants can be, e.g., team-leaders, lead developers or architects. In these meetings, the team members basically address the same issues as the team members do in their daily meetings, but at a higher level of abstraction. Typical issues that are discussed include what the team has done since the last meeting, what it plans to do, and whether it has or whether it plans to create any obstacles for the work of other teams. Things that typically might be regarded as obstacles include changes to an agreed interface, integration issues or architectural changes.

2.3.16 Encourage frequent communication

In addition to having status meetings often, the team members should be encouraged to communicate frequently also outside meetings. One should
aim at lowering communication barriers so as to make it as easy as possible
for people to contact whoever might be able to help them whenever they
have a problem or question. It is also important to introduce the persons,
who need to collaborate, to each other and encourage them to communicate.
We have noticed that frequent status meetings bring up issues which the
team members realize need to be discussed after the meeting in one-to-one
discussions or in small groups. Thus, frequent status meetings increase also
the amount of informal communication especially between the sites, which
is found to be very beneficial.

Perhaps the best way of lowering communication barriers is to have peo-
ples meet face-to-face, getting them to learn to know each other. Despite
all modern technology, it still seems that people need to meet, to feel fully
comfortable when contacting each other.

Another important aspect is to make available a set of collaboration tools
that can be used for communication. Different situations, as well as varying
personal preferences, make it important to have the possibility for both syn-
chronous and asynchronous communication, with varying degrees of media
richness. Thus, both text-based as well as audio- and videoconferencing sys-
tems should be provided. In many cases, it is also useful to provide technol-
gy that enables screen sharing, since that allows people from different sites
to work simultaneously on very technical tasks, e.g., debugging or coding.
Screen sharing is also useful for demonstration purposes.

2.3.17 Encourage open communication to locate and solve
problems quickly

Create a positive atmosphere in which it is easy for anyone to bring up any
problem or issue as soon as it emerges. The earlier an issue is raised the
lower the cost of resolving it. It is therefore important not to “shoot the
messenger”, i.e. punish someone for telling bad news. Instead, it should be
viewed as a service for the whole team if someone points out a problem.

A related matter is to plan for problem solving communication. Problem
solving communication is easily forgotten in project planning, even though
it is commonly needed in distributed projects, especially when facing a lot
of uncertainties, e.g., concerning new technologies. If channels for problem
solving communication are not agreed upon at the beginning of the project,
it might take a long time before problems are solved and delay the whole
project. When no suitable communication practices exist, project members
easily waste a lot of time trying to find a person who can help them. In ad-
dition, the barrier for off-site personnel to contact the on-site can be high,
even when faced with serious problems, if this has not been dealt with pre-
emptively. If problems are brought up late, they can be both difficult and expensive to solve. Thus, ensuring open and positive communication channels, and recognizing the need for efficient problem solving communication is important when planning and conducting distributed projects.

2.3.18 *Arrange frequent visits*

Building an efficiently working and communicating team is less painful if the team members can meet each other face-to-face at least in the beginning of the project. Preferably, the first visit should not be only a short trip to meetings, but a longer stay during which distributed team members can start working together on project tasks. Later on during the project, both short trips and collocated working periods are useful to maintain a good collaboration relationship, e.g. when planning a new sprint, when having problems, or for testing and integrating the product together before a major release. Even though you might have a good infrastructure for electronic communication, face-to-face meetings are needed to build a common understanding and an efficiently working and communicating team. Plan the travelling schedule in the beginning and remember travelling costs in your budget. Divide travelling responsibilities between your team members and sites. You can find more information about frequent visits in the next chapter (Chapter 3).

2.3.19 *Arrange collocated working periods*

Certain project phases or situations can be best solved by using periods of co-location. During such periods (a part of) the team travel and work at the same site as the rest of the project team. Co-located working periods seem to be particularly useful in the following situations:

- *When starting the project:* e.g., training new offsite members.
- *When planning new iterations:* even for a whole planning iteration.
- *When having problems,* both technical and social: in particular when facing motivational or other human-related issues.
- *For integration and testing:* e.g., when doing system testing before a new release – testers and developers can sit together in the same room and solve problems and fix bugs as soon as these are found.

2.3.20 *Use visiting engineers and managers*

Visiting engineers are project team members, often experts such as architects, who travel frequently between sites and stay and work at distant sites
CHAPTER 2. COMMUNICATION AND COLLABORATION . . .

for short periods. They convey information between the sites, solve problems together with remote team members, might arrange training or knowledge transfer sessions, help with taking new practices or processes into use, and create contacts between the sites. Visiting engineers, or liaison persons, might also stay and work at the remote site for a longer time period. In cases like that, one of their tasks could be that of a cultural liaison. You can read more about liaison persons in the previous chapter (Chapter 1).

For team members as well as for management, among them project managers and product owners, it is important to visit the remote site or sites frequently to create close contacts with the remote team members, observe the feelings at the remote site (e.g., problems might be communicated much later than they can be observed at the site), and show that the project, with every single team member is important. For example, in one of our case projects the product owner, with a lot of experience from distributed projects, visited the off-site regularly and took with him always one on-site expert that he felt was needed to help the off-site at that moment.

2.3.21 Create peer-to-peer links

To facilitate communication, the explicit forming of strong peer-to-peer links, e.g. between managers at the same level, developers and testers, as well as architects and other specialists who need to collaborate in the project is often worthwhile. This can be done by introducing them to each other, having them work on provided common tasks, providing them with an opportunity to meet face-to-face, or even for some periods of collocated work.

2.3.22 Arrange design and code reviews

While basically a technical practice, we have found that instituting peer-level design and code reviews can provide many benefits from a communication and trust building point of view. Early code and design reviews help ensure that all sites and team members involved have a common understanding of the task at hand. While it is easy to say “Yes, I understand” when asked in a meeting, explaining how something should be or is implemented is another matter entirely. It is important to understand that good design and code reviews are not supposed to be confrontational or blame seeking. In particular in the beginning of a project or partnership, these reviews might function as training, where the on-site personnel explain implementation details to their off-site peers to help them do a good job.
2.4 Summary

Table 2.1 summarizes the presented communication and collaboration practices.

Table 2.1: Summary of communication and collaboration practices

<table>
<thead>
<tr>
<th>Communication practice</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use frequent deliveries</td>
<td>Prefer short iterations, 2-4 weeks, and if possible use continuous integration and automated testing.</td>
</tr>
<tr>
<td>Reduce temporal distance</td>
<td>Choose partners preferably from the same or from a very close time-zone.</td>
</tr>
<tr>
<td>Reduce the number of sites</td>
<td>Do not distribute a project if you do not have to. Limit the number of sites to 2-3 at the maximum.</td>
</tr>
<tr>
<td>Reduce cultural distance</td>
<td>Prefer choosing partners from similar cultures or use cultural liaisons.</td>
</tr>
<tr>
<td>Define and communicate a clear organizational structure</td>
<td>Create clear organizational structure with responsibilities for the whole distributed project and communicate it to everybody.</td>
</tr>
<tr>
<td>Prefer dedicated team members</td>
<td>Choose team members who can work full-time for your project to make communication and coordination a bit easier.</td>
</tr>
<tr>
<td>Create one single team</td>
<td>Create a feeling of a single team – avoid division between “us” and “them”. Arrange common tasks, collocated periods, trainings, kick-off, etc.</td>
</tr>
<tr>
<td>Arrange face-to-face team building possibilities</td>
<td>Arrange face-to-face team building possibilities especially in the beginning of the project and between people who need to collaborate. Give “faces” to all sites and partners.</td>
</tr>
<tr>
<td>Provide background information</td>
<td>Provide information of the project’s goals, product roadmap, business case, customer’s business, etc.</td>
</tr>
<tr>
<td>Agree on communication practices and tools</td>
<td>Discuss and agree on, e.g., meeting schedules, raising issues, giving feedback, chat practices, use of telephone, email etiquette, ad-hoc meetings, communication tools, and development tools.</td>
</tr>
<tr>
<td>Agree on progress monitoring</td>
<td>Possible practices and tools: daily or weekly meetings, backlogs and burndown charts. Agree together the definition of “done”.</td>
</tr>
</tbody>
</table>
### Table 2.1: continued...

<table>
<thead>
<tr>
<th>Communication practice</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrange training</td>
<td>Arrange training, especially for off-site personnel if they come later into the project. E.g., class-room training, hands-on training and collocated working periods.</td>
</tr>
<tr>
<td>Create visibility across sites</td>
<td>Create visibility for team members about project progress, problems, decisions, changes, customer contacts etc. Possibility to see the “big picture”.</td>
</tr>
<tr>
<td>Arrange frequent status meetings</td>
<td>Arrange frequent status meetings. Daily is better than weekly. Have a clear agenda. Keep the meeting brief. Move discussions to after the meeting. Everyone should participate.</td>
</tr>
<tr>
<td>Arrange weekly meetings across teams</td>
<td>Arrange weekly meetings across teams. Have representatives from every team. Have a clear agenda.</td>
</tr>
<tr>
<td>Encourage frequent communication</td>
<td>Encourage team members to also communicate outside of the meetings. Introduce people who need to collaborate. Provide communication tools.</td>
</tr>
<tr>
<td>Encourage open communication to locate and solve problems quickly</td>
<td>Create a positive atmosphere to bring up problems or issues. Create efficient channels for problem solving.</td>
</tr>
<tr>
<td>Arrange frequent visits</td>
<td>Arrange frequent visits in the beginning to create contacts, later on during the project to maintain the relationships. Short trips and collocated working periods. Involve different persons from different sites.</td>
</tr>
<tr>
<td>Arrange collocated working periods</td>
<td>Arrange collocated working periods, e.g. when starting a project, for training, when planning new iterations, when having problems, and for integration and testing.</td>
</tr>
<tr>
<td>Use visiting engineers and managers</td>
<td>Use visiting engineers and managers to convey information between the sites, to solve problems together, to arrange trainings, to help with taking new practices or processes into use, and to create contacts between the sites.</td>
</tr>
<tr>
<td>Create peer-to-peer links</td>
<td>Create links between managers at the same level, developers and testers, architects and other specialists who need to collaborate. Introduce the people, create common tasks, provide possibilities to meet or even periods of collocated work.</td>
</tr>
<tr>
<td>Arrange design and code reviews</td>
<td>Arrange design and code reviews early in the project to ensure that all sites and team members have a common understanding of the tasks.</td>
</tr>
</tbody>
</table>
Chapter 3

Using Scrum Practices in GSD Projects

Maria Paasivaara and Casper Lassenius

3.1 Introduction

This chapter provides advice for applying Scrum practices to globally distributed software development projects. We discuss the use of distributed daily Scrums, Scrum-of-Scrums, sprints, sprint planning meetings, sprint demos, retrospective meetings, and backlogs. Moreover, we present a couple of lessons that distributed Scrum projects can learn from globally distributed software development projects: frequent visits and multiple communication modes.

The advice presented is based on a study of four globally distributed projects that were using Scrum. All projects were developing new software, either a new product, a new service, or a new version of a product. The largest project had seven Scrum teams distributed between two sites, while the smallest project consisted of a single distributed Scrum team. One of the projects had a co-located Scrum teams in each of the two sites. The development work in all the projects was distributed between two sites, which we call "on-site" and "off-site". By on-site, we mean the main site which holds project ownership. In the projects we studied, for two projects the on-site location was in Norway, and for the other two in Finland. In addition to project ownership, the on-site locations were responsible for customer contacts, and most domain and system experts were located at those sites. In each project the off-site location was in a country with cheaper labor: Malaysia, the Czech Republic, Russia, and Lithuania, respectively. We gathered data from the projects by interviewing project personnel on the
practices used, challenges encountered and successes achieved. Altogether we interviewed 24 persons. Of all projects, this was the first Scrum project for the team. The personnel in all teams were very happy about this change from a more traditional process model to Scrum.

3.2 Distributed daily Scrums

"I think that daily Scrum meetings were the best thing that happened to these distributed teams."

– A comment by a distributed Scrum team member.

The daily Scrum meeting is clearly the most important Scrum practice for distributed projects. In this daily team meeting, which lasts approximately 15 minutes, each team member answers the three Scrum questions: "What did you do since the last Scrum meeting? Did you come across any obstacles? What will you do before the next meeting?" After answering the questions, a brief discussion will take place. The purpose of the discussion is not to solve problems, but to decide e.g., who will need to discuss or solve the problems later on.

3.2.1 Application of daily Scrums to distributed projects

When your Scrum project team is distributed between two or several locations, you can arrange distributed daily Scrum meetings, e.g., by using videoconferencing. If a video connection is unavailable, a good quality voice connection will do, perhaps augmented with web cameras, or you can use only instant messaging. Arranging daily Scrums requires that there is at least some work time overlap between all participating locations. This is an important consideration when choosing the locations for the project. We think that using asynchronous daily Scrums, e.g. using only e-mail, is unlikely to work well.

Originally, daily Scrum meetings were designed to be arranged as face-to-face gatherings. In a distributed project this is not possible, but you can arrange circumstances that are as close to a face-to-face situation as possible. In the absence of a virtual presence solution, a good quality videoconference connection or even web cameras make it possible to recognize who is talking and to see facial expressions during the meeting. This makes the situation more natural, helps in creating joint understanding and building team spirit for the distributed team.
CHAPTER 3. USING SCRUM PRACTICES IN GSD PROJECTS

If your project has several distributed teams that all have their daily Scrum meetings, you can follow the example of one of our case projects. In this project the teams, distributed between two locations, had consecutive daily Scrum meetings. The 15 minute long meetings took place in the same meeting room, one after another. Thus, the connection, with voice and web cameras had to be set up only once.

If video- or teleconferencing is not possible because of technological or other issues, such as problems with spoken language, using chat is an option. Some people might feel more comfortable writing instead of speaking due to (subjective) difficulties with pronunciation or understanding spoken foreign language. One of our case projects used internet relay chat (IRC) for arranging daily Scrum meetings. Typically, all team members wrote their answers to the three Scrum questions prior to the meeting, and the meeting commenced by everybody sending their answers, and reading the others’ messages. Subsequently, a discussion took place. Chat logs were saved for those who were not able to participate. When we conducted our last interviews, however, the company involved had acquired videoconferencing equipment and the team was planning to start using videoconferencing at least once a week for the daily Scrum meetings, thus abandoning their reliance upon text-only daily Scrums. Based upon their experiences, it seems that using only chat for daily Scrums is possible but not to be recommended, as much information is lost compared to tele- and videoconferencing. For example, in a teleconference the tone of voice often relays important but non-explicit information, as do facial expressions seen when using virtual presence or videoconferencing systems.

If your project has site-specific Scrum teams, the teams can have normal face-to-face Scrum meetings. However, it is important to share information frequently between the teams at different sites. One possibility is to use Scrum-of-Scrum meetings for sharing information between the teams, as explained later in this chapter. In the case of just a couple of teams, one team member representing each team can participate in the other Scrum teams’ daily Scrum meetings every day or a few times a week to share information between the teams.

3.2.2 Benefits of daily Scrums

Our interviewees reported that distributed daily Scrum meetings were the most useful practice for distributed projects. The benefits of daily Scrums are numerous: they provide frequent possibilities to share information and coordinate work between distributed team members, they help to recognize possible problems early on, they provide a possibility to create contacts, and
CHAPTER 3. USING SCRUM PRACTICES IN GSD PROJECTS

they encourage team members from different sites to communicate more actively, also facilitating off-line communication after the meetings.

Daily Scrum meetings provide a good way for everybody in the distributed team to get an overview of the project situation. In particular, our interviewees reported that it was easier to monitor the offshore situation than before. Moreover, daily Scrums help to identify problems quickly, since with daily monitoring it is difficult to hide problems over a long period of time.

When problems or a need for one-to-one discussion are encountered during daily Scrums, the teams should set up separate meetings after the daily Scrums and continue discussions in smaller groups or one-to-one either by video-, or teleconference, chat or email. In all our case projects, daily Scrum meetings encouraged team members to communicate more also outside the meetings, which was seen as one of the greatest benefits of these meetings.

3.2.3 Challenges of daily Scrums

Even though there are numerous benefits in arranging distributed daily Scrum meetings, there are also some challenges. The biggest challenge for distributed teams is the same as for co-located teams: understanding what the correct amount of information to report in a Daily Scrum meeting is. This is challenging even in a co-located project, but in a distributed project it is even more difficult. The team members do not know what others find interesting or important. Thus, the team needs to practice this with the help of their Scrum master. In one of our case projects, the daily Scrum meetings initially lasted only a few minutes, before the team members learned to discuss actively, and in particular to be open about their impediments. The Scrum masters started to encourage everybody to talk and share more about their tasks and impediments. Thus, the teams ended up having 15-minute meetings that were found very useful by all the participants.

Cultural differences may have a big impact on what people find appropriate to report in a daily Scrum meeting. We noticed that there are huge cultural differences in revealing impediments and discussing them in daily Scrum meetings. For example, in Scandinavian cultures talking about impediments is much more natural than in Asian cultures. Moreover, when team members come from different companies, they are more likely to hide problems, in particular in the beginning of a project.

When comparing projects that had distributed daily meetings to projects with mainly site-specific daily meetings, a clear difference could be seen. Most of the participants of the distributed meetings mentioned the benefits: increased transparency to the other site, getting a good overview of what was happening in the project, and well functioning and open com-
munication across the sites. However, the participants of the site-specific, non-distributed meetings talked about problems: they did not have enough communication and contacts with the other site, nor did they know enough what was happening at the other site. Thus, it is important to share information also between Scrum teams in the same project. Especially when the teams are site-specific, informal communication between the teams does not occur naturally, i.e., the way it does when around a coffee table. We will discuss more about sharing information between Scrum teams later on, in the section on Scrum-of-Scrums.

---

**Daily Scrums - tips:**

- Provide a good infrastructure for daily Scrums. Meetings should be easy to set up and provide as rich communication facilities as possible: virtual reality systems or videoconferencing are the best. If unavailable, a good quality voice connection will do, perhaps augmented with web cameras. Use text-only meetings only as a last resort. Avoid asynchronous "meetings".

- Work actively with the team by practicing and discussing, to find the optimum type and amount of information to report in the daily Scrum meetings.

- Create an open atmosphere that makes it easy to raise problems and issues without fear.

- Encourage discussions in small groups or one-on-one after the daily Scrum meetings and arrange a technologically good infrastructure for these distributed discussions.

---

### 3.3 Scrum-of-Scrums meetings

Distributed projects that have two or more Scrum teams need to share information between the teams. One way to share information, mentioned earlier, is to have one team member participating in the other teams’ daily Scrum meetings. This is practical only when there are no more than two or three teams. When the number of teams increases, arranging Scrum-of-Srum meetings is a must.
CHAPTER 3. USING SCRUM PRACTICES IN GSD PROJECTS

The objective of Scrum-of-Scrum meetings is to share information between teams regarding what is happening in the teams, what kind of challenges the teams are facing, and what kind of interconnections the work done by different teams has. Scrum-of-Scrum meetings provide good possibilities to create contacts and encourage communication between the teams.

One team member from each team participates in the Scrum-of-Scrum meeting as a representative of his or her team. The team decides who is to represent them; the participant does not always have to be the same person. In addition to the team representatives, in one of our case projects all Scrum masters participated in these meetings.

Scrum-of-Scrum meetings normally take place once a week, but they can also be arranged more frequently if there is a more frequent need for coordination between the teams. A suitable length for a weekly Scrum-of-Scrum meeting is half an hour.

During the meeting the three Scrum questions are answered, at the level of the team, however. Thus, each team representative explains what his or her team has been doing since the last meeting, what the team is planning to do before the next meeting and what kind of impediments the team has had. Moreover, you can have two additional questions: "Have you put some impediments in the other teams’ way?" and "Do you plan to put any impediments in the other teams’ way?"

3.3.1 Application of Scrum-of-Scrums to distributed projects

Scrum-of-Scrum meetings can be applied to distributed projects in a similar way to daily Scrum meetings. The only difference to co-located projects is the need to arrange the meeting virtually. The same technologies as used in daily Scrum meetings can be used. Good quality videoconferencing provides the possibility to easily recognize who is talking and to see facial expressions. This is important since at least some of the participants may not have met each other face-to-face. In the absence of videoconferencing, web cameras are helpful in supporting a teleconference call.

3.3.2 Benefits of Scrums-of-Scrums

The Scrum-of-scrums meetings distribute information between the teams and reveal possible problems early on. They open discussion channels between the teams and that way encourage informal communication. One of our case projects, which used weekly Scrum-of-Scrum meetings to coordinate actions between their seven Scrum teams, felt that these meetings were very beneficial. On the other hand, a project having two site-specific teams
at different locations, but not using Scrum-of-Scrum meetings, mentioned several problems: they did not have enough communication and contacts with the other site, nor did they know enough about what was happening at the other site — exactly the problems that Scrum-of-Scrum meetings are designed to prevent!

### 3.3.3 Challenges of Scrums-of-Scrums

The challenges in arranging distributed Scrum-of-Scrums are the same as for distributed daily Scrums: finding a suitable level of reporting that is both useful and understandable for all parties, cultural challenges and trust issues in reporting impediments, and forwarding the important information to the rest of the team members.

**Scrum-of-Scrums - tips:**

- Practice and talk with the participants about the correct type and amount of information to report in Scrum-of-Scrum meetings.

- Create an open atmosphere that makes it easy to discuss problems and issues.

- Encourage discussions in small groups or one-on-one after the Scrum-of-Scrum meetings and build a good technological infrastructure for such discussions.

- Provide a technically good and easy to set up infrastructure for the Scrum-of-Scrum meetings: a good quality voice connection, preferably also video, to be able to recognize who is talking and to see facial expressions.
CHAPTER 3. USING SCRUM PRACTICES IN GSD PROJECTS

3.4 Sprints

”Before [we started to use] Scrum I could not really understand when there is a deadline and what should be done by that deadline [...] because there were many different deadlines for customers and development stages [...]”

– A comment by a distributed Scrum team member.

Iterations in Scrum are called sprints. The length of one sprint in Scrum is normally from one to four weeks. Our case projects used both four-week and two-week sprints.

If a project has several teams, it is a good idea to synchronize the sprints, i.e., have all teams start and end their sprints at the same time. For example, a large case project had synchronized 4-week sprints in the development teams. The maximum variation of the end and start dates was a couple of days. The same project also involved a maintenance team that had a sprint cycle of only two weeks synchronized with other teams’ four-week cycle. The reason for this shorter cycle for the maintenance team was to make them able to release fixes to customers every two weeks. This system of synchronized four-week and two-week sprints worked well according to our interviewees.

3.4.1 Application of sprints to distributed projects

Using sprints in a distributed project does not differ much from using them in a co-located project. The sprint length in co-located projects can sometimes be as short as one week, but distributed projects make the required meetings more cumbersome, in particular when it comes to planning, demos and retrospective meetings, so we think that a two-week sprint length is a good minimum sprint length in a distributed project.

The sprint lengths of different site-specific teams should be the same, but there can be exceptions. For example, in one of our case projects the sprint length of the on-site team was four weeks, while the off-site team had sprint duration of only two weeks. The shorter sprint length at the off-site made it possible for the on-site to better support the off-site team.

Different vacation times in different counties may pose challenges, as team members might be unavailable for a substantial part of a sprint. Sprints might be lengthened to keep sprint content reasonable and the sprints of different teams synchronized.

In the beginning of a distributed project it is beneficial to arrange face-
to-face meetings for all team members, so that everybody can at least once meet and learn to know each other. One approach is to invite the whole team to work together in a single location for one or two sprints. This way the team can build a common understanding of the project goals and learn how to work together. Team members also have a chance to get to know each other. After such a co-located period, it is a lot easier to work in a distributed manner. Co-location can be a good idea also when testing and fixing the software, e.g., during the last sprint before a critical release.

One of our case projects used co-located sprints on a need basis. In this project, especially team members from the off-site travelled to the on-site location. The visits normally lasted between two and four weeks, which made it possible for the team members to really work together. In particular during critical phases, it was considered important to co-locate the team, e.g. for the last sprint before a release or for the first sprint in a new release project, when most of the planning took place.

3.4.2 Benefits of sprints

Short sprints hugely increase the transparency of distributed projects. Sprints with clear deadlines and goals make it easier for all team members to understand what is supposed to be done during the next sprint. In particular, team members at off-site locations benefit a lot, since often they do not have a clear picture of the overall project in a traditional distributed setting.

In addition, the frequency of feedback between on-site and off-site is increased. There is no possibility to delay the completion of a task because it is ”only 95% ready”. Moreover, short sprints reveal quickly, e.g., whether off-site personnel have misunderstood the requirements, and the problem can be solved immediately.

Finally, short sprints, with frequent regular meetings, make it easier for a distributed team to create a joint team identity, making the members feel like they are on a single team working for common goals, rather than being on two or several separate teams, not really understanding each other.

3.4.3 Challenges of sprints

The main risk with planning on a sprint-by-sprint basis is losing track of the ”big picture”, i.e., the overall goal of the project. Keeping the overall goal in mind is important, not least from the point of view of the resulting product architecture.

The planning overhead, involved in each sprint, can tempt one to use too long sprints, in which case their benefits erode.
Sprints - tips:
- Synchronize sprints between teams
- Do not have sprints that are shorter than two weeks in a distributed project
- Arrange co-located sprints when starting a project or facing challenges so that the whole distributed team can work together

3.5 Sprint planning meetings

At the beginning of each sprint, teams hold a sprint planning meeting. In the meeting, the backlog items to be developed in the sprint are selected, broken down to tasks, and the effort needed is estimated. The product owner presents and explains to the team the backlog items and answers the team members’ questions. Then the team together plans the sprint.

3.5.1 Application of sprint planning meetings to distributed projects

If a Scrum team is distributed, the sprint planning sessions can be arranged as distributed meetings.

However, if possible, it can be a good idea to invite all distributed team members to a single location to plan the next sprint face-to-face. If the team members are not too far apart, this can be arranged regularly. You can also consider arranging co-located sprint planning meetings at least for the first or first few sprints. That makes it possible for the team members to meet at least once face-to-face, to get to know each other. Unfortunately, in most cases arranging face-to-face meetings in two-week or four-week intervals is not economically feasible.

In one of our case projects, where the on-site and off-site locations were located at a reasonable distance from each other — a one-hour plane trip, that is — a couple of off-site team members flew to the on-site location for the first few sprint planning meetings, which made these meetings more efficient. After this good start, the meetings were arranged in a distributed manner supported by teleconference and application sharing.
Another possibility is to divide the sprint planning meeting into parts: co-located meetings at the different sites and a common distributed meeting for the whole team. In a large case project, with only three hours synchronous working time between the on-site and off-site locations, the sprint planning meetings were divided into three parts: a distributed meeting, a local meeting at the on-site location, and a local meeting at the off-site location. The distributed meeting was arranged using teleconferencing and application sharing. During the distributed part, the product owner presented the prioritized items in the backlog, and the team asked questions. Because of the time-zone difference this part of the meeting was time-boxed for the three common working hours for both sites. After the meeting, the off-site working day ended. The on-site team continued by dividing the backlog items into more detailed tasks, adjusting the estimates made by the product owner and making initial assignments of the tasks to different team members. The off-site team continued the work the following morning by discussing and commenting on the draft plan they had received from the on-site.

If a project has site-specific co-located teams, sprint planning meetings can often be arranged face-to-face. However, even if the team is co-located, the product owner may be located at another site, introducing the need to arrange at least a part of the meeting virtually.

### 3.5.2 Benefits of sprint planning meetings

Sprint planning meetings provide team members with an opportunity to participate in planning and thus both better understand what is expected of them, and to commit to the plans. In a distributed team, these meetings provide visibility to the work on both sites and offer a regular discussion forum. Sprint planning meetings also provide opportunities for building team cohesion and identification, despite of distribution.

### 3.5.3 Challenges of sprint planning meetings

Arranging a distributed meeting is always a challenge. All our case projects found co-located planning meetings preferable. However, despite the fact that all case projects had positive experiences with co-located meetings, arranging them regularly proved too expensive. Project members commented that planning is a challenging task that requires lots of discussion, which is difficult to do efficiently while planning is distributed. Also, the issues discussed are sometimes just difficult to explain in the distributed setup.

Long distributed meetings can be also very tiring, if, e.g. the voice connection between the sites is not very good — a situation that was not uncom-
mon to the team members we interviewed. Moreover, if videoconferencing or web cameras are not used, it can be difficult to know who is talking when not seeing the persons from the other site.

You have to take into account cultural, as well as knowledge differences between the sites. Otherwise, experienced developers at on-site might end up doing the planning with off-site developers just listening and not actively participating.

### Sprint planning meetings - tips:

- If possible, plan visits so that sprint planning meetings can at least sometimes be arranged face-to-face
- Encourage all team members to participate actively in planning
- Ensure technically good circumstances for virtual meetings: a quality voice connection, a working video connection if possible, and application sharing

### 3.6 Sprint demos

At the end of a sprint, the team demonstrates the developed functionality to all interested parties. The meeting is called a sprint demo or sprint review meeting.

#### 3.6.1 Application of sprint demos to distributed projects

In a distributed project, demos are normally arranged in a distributed manner. Even though the project team might be co-located, there are often parties, such as the product owner or team members from other teams, who are interested in participating in demos from other sites. If your project has just a couple of teams, they can have joint demos. This allows the teams to give and receive immediate feedback.

All our case projects arranged demos in which both on-site and off-site personnel participated. The demos were normally arranged using teleconference and application sharing. During visits, face-to-face demos were sometimes arranged.
CHAPTER 3. USING SCRUM PRACTICES IN GSD PROJECTS

3.6.2 Benefits of sprint demos

The demos increase the visibility of the project to all participants of the demo, especially between the distributed sites. They also offer a possibility to give and receive feedback, as well as to monitor the work at off-site. For example, in one of our case projects, before starting to use Scrum, the on-site and off-site teams were working independently for long periods of time. This commonly led to a lot of rework for the off-site team, as they often misunderstood the requirements written by the on-site personnel. Short sprints with demos at the end mitigated this problem.

3.6.3 Challenges of sprint demos

The biggest problem with demos is often the same as with other distributed meetings: the technology does not offer good enough possibilities to communicate efficiently. Our case projects typically used teleconferencing with application sharing to arrange their demos, but they were not happy with this technology.

Sprint demos - tips:

- In a multi-team project, invite also members from other teams to the demo and provide for a possibility to participate in a demo also virtually; even though the team might be co-located there can be interested parties from other sites. A demo provides a good possibility to share information, and to give and receive feedback

- Ensure technically good circumstances for virtual meetings: a high quality voice connection, video connection if possible, and application sharing

3.7 Retrospective meetings

A retrospective meeting normally takes place at the end of a sprint. During that meeting the team discusses three questions: "What has been good during this sprint?", "What has not been that good?" and "What kind of improvements could we do?"
3.7.1 Application of retrospective meetings to distributed projects

Retrospective meetings in distributed Scrum projects can be arranged in ways similar to the planning meetings. In particular, it is important to create an open atmosphere in which everybody’s input is welcome and valued. In particular, people at the on-site location should try not to dominate the meeting too much.

Technically, the meeting can be done using tele- or videoconferencing, and perhaps application sharing to jointly write the minutes.

In two of our case projects, the retrospective meetings took place as distributed teleconference meetings directly after the demos.

3.7.2 Benefits of retrospective meetings

The main benefit of the retrospective meeting is that it provides a time in which the whole team reflects upon its own behavior, and on how to improve it. Conducted successfully, the meeting can also provide good opportunities for increased team identification and commitment.

3.7.3 Challenges of retrospective meetings

In order for retrospective meetings to be successful, it is important that everybody participates and tries to contribute. Dominant personalities and experts should make a special point of recognizing contributions from less active or more distant members. As an example of a suboptimal practice, one case project with several Scrum teams had retrospective meetings that consisted only of the Scrum masters. While this can be useful as such, it is no substitute for meetings that involve the whole team.

Retrospective meetings - tips:

- Make sure that the whole team participates actively in the retrospective meetings
- Create a positive and open atmosphere that makes it easy to participate; recognize even small contributions
- Be sure to follow up on the issues raised and suggestions presented in the meeting
3.8 Backlogs

Backlogs are lists of items, e.g., features, to be developed. In the sprint planning meeting, the product owner, with his or her team, selects items from the product backlog to be developed during the next sprint. The features are then broken down into tasks that are estimated and placed in the sprint backlog. There are several commercial and open source tools for managing backlogs.

3.8.1 Application of backlogs to distributed projects

Co-located Scrum teams may manage their backlogs using physical objects, such as post-it notes on the wall. In a distributed team, this is not practical, since all team members, as well as their product owner, need to get access to the backlog. Thus, electronic tools are needed.

Our case projects used different tools to manage their backlogs, e.g. Wiki was used by a small project, whereas Jira was used by a large project. There are also specific backlog management tools available, such as Scrumworks and an open source tool Agilefant, but none of our case projects used any of them.

How the backlog is managed needs to be decided jointly with the team, the responsibilities for backlog management clearly assigned. In particular, the responsibilities of the product owner are critical.

3.8.2 Benefits of backlogs

Electronic backlogs with up-to-date information and access by all team members are necessary for managing tasks and monitoring progress in a distributed Scrum project.

3.8.3 Challenges of backlogs

The biggest challenges related to the backlogs in our case projects were related to unclear updating responsibilities. Especially the responsibilities of the product owner were quite unclear for new people assuming that role. In some projects, also the Scrum masters, chief designers or chief architects performed some of the product owner’s responsibilities. Moreover, in some projects there were several product owners who had not clearly divided responsibilities between them.
3.9 Frequent visits

In addition to learning how to apply Scrum practices to distributed projects, distributed Scrum teams need to take into account lessons learned from managing distributed software development projects in general. One important lesson is to arrange visits for distributed team members frequently enough.

3.9.1 First visit

Building an efficiently working and communicating team is less painful if the team members can meet each other face-to-face at least in the beginning of the project. Preferably, the first visit should not be only a short trip to a meeting, but a longer stay during which the distributed team members can start working together on project tasks. The length of a co-located working period could be, for example, one or two sprints. During this face-to-face period the team members learn to know each other and develop joint working habits by working together at least for a short period of time. This is an efficient start-up for a project and makes it easier to communicate and collaborate later on when team members are working from different sites.

3.9.2 Further visits

Later on during the project, both short trips and co-located working periods are useful. A co-located working period can be scheduled, for example, for a critical project phase like the last sprint before a release or for the first sprint of a new release when most of the planning takes place. It is a good idea to schedule the short trips so that the visitors can participate in the regular meetings face-to-face, making the meetings more efficient. Thus, it is ideal to schedule trips at the end of a sprint, so that the visitors can participate in the sprint demo, retrospective meeting and sprint planning meeting for the next sprint during the same trip.

Backlogs - tips:

- Choose a tool suitable for your purposes and give access to all team members
- Agree on updating responsibilities
In the beginning, when arranging the first meeting or a co-located working period for a new team, it is important that the whole team can participate. Later on, when arranging co-located working periods or short trips, the whole team does not necessarily need to travel. Instead, a few team members at a time can spend time at a remote site, on a need basis. However, it is important that it is not always the same persons who travel, but that every team member gets his or her turn. You can, for example, create a travelling schedule for the project. Moreover, it is a good idea to arrange trips to all sites, so that team members will get to know the circumstances at different sites and at the same time learn more about the local culture of their team mates. This way the task of travelling can be divided more evenly between the team members.

When planning a travelling schedule for your project, you can plan a regular schedule, for example a short trip every second sprint, or base your plan on the critical phases of your project schedule. Moreover, you probably have to arrange trips on a need basis. When your project is facing challenges, they are often easiest to solve face-to-face.

All our case projects arranged visits between the sites for team members, either on a need basis, or according to a regular schedule. The visits also included leisure activities, such as sauna or dinner. These gave team members a good possibility to get to know each other on a personal level.

3.9.3 Benefits of frequent visits

Frequent visits provide good opportunities for getting to know persons from the other sites, discuss difficult issues, and get a better picture of the project. Face-to-face meetings also increase trust between team members and encourage them to continue communication after the visits. It is important to arrange visits not only in the beginning of the project, but also during the project. All our case projects found their current model of frequent visits as very useful and even more visits were hoped for.

3.9.4 Challenges of frequent visits

Travelling comes with a high cost both in working time and money, thus it is important to plan the trips carefully. Getting travel plans accepted by higher-level managers is often thought to be the most challenging part of frequent visits. Motivating managers about the need to travel can be difficult: they might not appreciate the importance of meeting face-to-face and working together to build the team. Explaining that a trip will pay itself back quickly in better communication and in more efficient teamwork might not convince
them, since management might expect you to do perfectly well without the team having a chance to meet face-to-face! However, in our case projects this problem was never mentioned. Even though the case projects arranged quite a few trips, none of the interviewees mentioned any problems with trip arrangements due to cost.

In our case projects, mainly off-site personnel did the travelling. The reason for this was that on-site personnel included mainly experts who did not have time to travel, even though that was hoped for by the people at the off-site locations. Off-site personnel were mainly developers, who found it extremely useful to meet the on-site experts face-to-face and ask questions and discuss difficult issues. Our interviewees, especially from the off-site, hoped that on-site personnel would travel more to provide opportunities for additional off-site persons to meet them and to share the sometimes quite heavy and tiring traveling duties between on-site and off-site.

Finally, one challenge related to frequent visits, especially in arranging co-located working periods, is limited office space. Quite often this problem can be solved by planning ahead, e.g., by reserving a big enough team room or reserving a meeting room for the time of a co-located period.

**Frequent visits - tips:**

- Even though you might have a good infrastructure for electronic communication, face-to-face meetings are needed to build a common understanding and an efficiently working and communicating team

- Start a project preferably by a co-located sprint

- Plan the travelling schedule in the beginning and remember travelling costs in your budget

- Divide traveling responsibilities between your team members and sites

### 3.10 Multiple communication modes

In addition to face-to-face discussions, members of distributed Scrum teams need to communicate a lot electronically. Providing several good tools for different kinds of communication purposes is another lesson learned from
managing distributed software development projects. Different people, contexts and situations require different communication tools. The minimum set of tools that should be provided include:

- email
- instant messaging
- unrestricted voice calls
- application sharing

In addition, web- and videoconference solutions should be made available if possible. Often, videoconferencing equipment is a scarce resource. Optimally, one videoconference or telepresence room could be made available for spontaneous short meetings only, e.g. it cannot be reserved or used for hours by a single meeting.

In our case projects, tool choice seemed to depend both on the purpose of the communication (e.g., chat was used to ask short questions or for checking whether the other party was available to receive a phone call), as well as the preferences of the user. Some people preferred synchronous voice communication while others with limited language skills preferred written communication.

### 3.10.1 Benefits of multiple communication modes

The main benefit of allowing and providing for multiple communication modes is that it lowers the barriers to communication by allowing team members to communicate in a way that fits them best outside project meetings. Since poor communication or the lack of communication altogether is a common problem in distributed projects, one should not underestimate the importance of this.

### 3.10.2 Challenges of multiple communication modes

Providing for multiple ways of communicating is in principle easy, but corporate policies and IT departments sometimes make it unnecessary difficult. Try to find a way of providing for, in addition to email, at least the possibility for instant messaging and unrestricted voice communication between team members. For voice, an IP-based solution can help mitigate the fear of otherwise high phone costs.
### Multiple communication modes - tips:

- Aim at providing a rich set of communication tools that personnel can use also outside official meetings.

- Allow people to use the media they like the best with the least possible limitations. For traceability or other reasons, documentation can be done after an informal exchange, e.g. by email.
Part II

Teams and Individuals
Chapter 4

Trust and Distrust in GSD Projects

Arttu Piri

4.1 Introduction

This part of the book deals with the issues related to teams and individuals in the context of globally distributed software development. Most of the ideas presented here are definitely applicable also in the context of collocated software development. However, from the point of view of individuals who are working in software development teams, we believe that a distributed way to organize the development work certainly brings in some extra flavor compared to projects where people are physically located in the same place.

Development of commercial software products and client-specific projects is highly knowledge-intensive, fast-paced and complex. Clients’ unique needs in software projects and the general unavailability of routine solutions, with which software producers would be able to build products that meet the clients’ needs, are factors that will ensure that the level of complexity in software development will remain high. According to some estimates, nearly half of software development projects fail or are severely challenged what comes to their profitability, client/end user satisfaction or, for example, to their ability to keep the promised deadlines. Compared to some other areas of industrial production, the amount numbers is distinctively high.

In software development, the potential for optimal as well as the risks for non-optimal results lie ultimately in the individual capabilities and competences of specialized professionals involved in the development project. Building functionalities of software products typically includes writing the source code, which is usually done alone by a software developer and re-
requires individual creativity and coding skills.

But before the developer is able to apply his/her personal coding skills to build the needed features(functionalities), there is a large amount of work to be done, which cannot be accomplished alone: he/she needs to interact actively with the other members of the team and be able to digest the information shared by others and created together and related to the given product, feature or functionality. Similar demands for flexibility between working alone and dynamic collaboration with colleagues is needed in other areas of software development.

So it seems that one of the success factors, and also the challenges, in software development is how to utilize the internal creativity and problem solving skills that the software professionals have, but also maintain a culture of collaboration, open discussions and interactions in the software teams. Complete products or projects are being constructed by sequentially tying up the individual contributions for the project to a single entity. Thus, the potential success in software development can be described as a result of a combination of individual creativity together with their capabilities to produce quality code, and the abilities of these individuals to work as a team and to create a solid and shared understanding of the product by negotiating and sharing information and interacting with the others.

In this chapter we emphasize the importance of solid collaboration, healthy group relations and productive team work as components that are often crucial for successful software development. Although there is certainly a place for individual problem solving and introvert creativity, software development projects can generally be understood as collaborative endeavors. Following this, all the issues that harm software development teams’ abilities of collaboration are potential risks from the viewpoint of project success. Next we discuss some specific types of problems related to GSD teams found from cases in the MaPIT project.

4.2 Problems related to group relations in GSD

By going through the data from eight GSD projects, we identified the most common problems related to group relations in these projects. The most important problems (not in any specific order here) were:

- Lack of trust between employees at on-site and off-site locations

- Concerns related to one’s own position and responsibilities in the future at on-site, due to a new distributed project organization (fear)
CHAPTER 4. TRUST AND DISTRUST IN GSD PROJECTS

- Weak team cohesion and disintegration between the on-site and off-site teams

In the following part of this section we will discuss teams and individuals in GSD from the point of view of the problems identified above. They illustrate how personal thoughts, feelings, emotions, and attitudes of the members of globally distributed development teams strongly affect the way of collaboration in a project. In short, a distributed team’s ability to efficiently work together towards the project’s goals is based on single individuals’ motivation, which is tied to their general understanding about the nature and context of collaboration and the relationships between the individuals in the team.

4.3 What trust means?

Trust is a bit special concept, because it has formal scientific definitions and, at the same time, it is a noun with a generally agreed meaning and a word that is used all the time in everyday communication. Most of people have an idea about what ”trust” and ”distrust” basically mean, at least according to their own understanding.

In the research of the ”soft” or ”human” issues in GSD, in other words in research focusing on the social relations and group dynamics, trust and lack of trust between employees of distributed projects are by far the most studied topics. It has been widely understood that a certain level of trust between employees working in the same development team is needed to get them cooperate properly, and that distrust between people usually makes collaboration more difficult, often causing some notable problems also at the project level. In general, these findings are strongly supported by the MaPIT data.

Several more or less exact definitions — mainly of academic interests — aimed to capture the complex nature of ”trust” have been introduced. Without going any deeper into them, we follow the definition\(^1\), that trust can be described as a belief of a person (trustor) that some other person or group (trusted or trustee) will not act ”opportunistically” or use the vulnerable state of the trustor in a manner that purposefully causes harm for the trustor in some particular situation(s) or context. Opportunism means here that the trusted party purposefully tries to gain some personal advantage of the situation and is not aligned with or contradicts the trustor's expectations regarding the trustee’s behavior and outcomes of the action.

\(^1\)by Mayer et al. (1995)
4.4 Trust in GSD teams

In a context of real-life software development teams, according to the MaPIT interview data, experiences of trust and distrust are usually related to the level of confidence on whether the GSD team members are technically competent to efficiently take care their responsibilities in the team, and whether they as workers are as motivated and capable of pushing the project forward as strongly as are the people in the project in general. So feelings of trust and distrust have a strong link to expectations that individuals set for the behavior or performance of other people, in this case their colleagues, subordinates or supervisors in a distributed software development team.

Case example:

The next example from one of the MaPIT-case projects describes a team leader’s trust on his team members located off-site.

Q: And now I’m throwing it back to you: what do you mean by trust in this business relationship? What does it mean to you?

A: I know that when I, if I have got information from them that we will manage to do this, I know that they will really try to do it. They will work night and day to fix the thing. If I say to them that you have to be ready by Friday, ”okay, I will try to do it”. And they really try, I trust that they will work every hour. They don’t waste time, they will not stand by the coffee machine and chat for an hour or two, they really work. So they are really, yeah, involved and they want to deliver and they are really clever.

But even before being able to evaluate the trustworthiness of people based on their behavior, for example in a recruiting situation or in the early phases of a GSD project when we don’t yet have any actual experiences of other people’s behavior, we try to form an opinion about them, among other things about the level of their trustworthiness. We do this, either unconsciously or consciously by looking for suitable cues on which we could ground our estimations and assumptions about the probable trustworthiness of the individual. It is highly personal what are the cues that might act as ”triggers” for trust or distrust, but often they can be related to factors like cultural or ethinical background, demographic information, education or social skills and presence.

Following the process called social categorization, which means that we try to simplify our perception of our social environment by positioning in-
individuals to different categories based on various factors, we often tend to think that people that we have categorized as more similar with us are more trustworthy or at least somehow more optimal people to collaborate with compared to individuals who we think are very different from us. When joint collaboration has advanced a bit further, for example when a new GSD team moves from a project’s kick-off phase to the normal development rhythm, it is possible to start reflecting the assumed trustworthiness of the people with their actual behavior. With a similar logic, trust or distrust can be directed towards a particular individual (i.e. some particular colleague), or towards a larger group or community.

Case example:
The next example illustrates how, after a difficult start in a project, the trust towards off-site team members was developed by giving then opportunities to show their competences.

Q: How did you start, what happened? You say that now you trust them, what has happened? Why do you trust them now?

A: It has been a long way. We have had this training, if you can call it a program, but we have had some training periods. We have had this development task that they solved very satisfactory, yeah. It’s a good solution that they have made, and we can see now that they are really good at what they are doing, yeah. And we also have had several bug fixings that they have solved for us, and we can see that they have solved things that we haven’t managed to solve up here, because we haven’t the right skills. So, they are good.

Trust cannot be intentionally ”built” by external actions or with components or managed to take effectively place in some situations. Trust may be developed in the interactions between the people involved, and is rooted to the fulfillment of the expectations these people have set for these interactions.
4.5 Is the question about trust or competences?

The importance of trust is probably easier to realize when there is no trust, or when trust is tested or questioned, than when there aren’t any problems with trust issues. When trust is high among the team members, it probably feels so natural and self-evident that it won’t get any special attention. But in general, teams with high trust often share some common characteristics.

In software development, things are rarely going as smoothly as initially planned. All projects will face problems. Some of the problems can of course be traced back to for example clients changing their minds or not really knowing what kinds of functionalities their software should have. But there are always also other sources of problems; the source code turns out to be buggy, some of the developers are faster and able to produce better quality code than the others, tasks turn out to be far more complicated than was estimated, immature technological components and solution have been chosen for the product or project, and so on.

All these problems are common in software development because of people’s general tendency to make mistakes and false evaluations with difficult tasks and decisions. Industrial software development is highly knowledge-intensive work of special type in one of the most complex forms possible. It deals with abstract constructs, with which both creativity and structured approach as well as capabilities for teamwork and independent work are needed. But what has this got to do with trust?

In trustful GSD teams, normal project shortcomings will not generally cause concern about the reliability of the employees. Social relations, if there is trust, are flexible also in situations where problems occur. Problems won’t become personalized and will be solved without having effect on the future expectations towards the behavior or the person or group in question. If an analysis of the reasons for some problem or failure indicates a crucial shortage of competences and skills, it may create an issue to be solved. But what we have found out in our studies of GSD teams is, that some limitations in personal competencies are often paralleled with the level of trustworthiness of the person in question.

From the theoretical point of view this kind of thinking is very interesting: if we assume that people behave sincerely, it is hard to understand why there should be a link between making mistakes and trustfulness. Why should this be an issue of trust? Shouldn’t the level of competences and the corresponding performance of an individual be treated as such, with no connection to the level of his/her sincerity?
CHAPTER 4. TRUST AND DISTRUST IN GSD PROJECTS

Case example:
The next quotation from an interview shows how distrust towards team members at off-site was based on unawareness of their competences.

Q: What about trust in the beginning of the relationship? Did you trust them from the start?
A: No, not that much, because I didn’t know them, I didn’t know what kind of people we got down there, whether they would be educated or experienced or what we could expect from them. So I don’t, I can’t say that I didn’t trust them but I don’t know if I should trust them or not, because I haven’t any experience of outsourcing and I don’t know them and I don’t . . . I knew that we have a difficult area down there.

4.6 How to recognize a GSD team with high trust?

In trustworthy teams information sharing is most likely open and unrestricted. Information sharing and exchange is not ruled by considerations of who needs exactly what information and what are the risks if some information gets to colleagues who don’t seem to necessarily need it in their work. Positive consequences (ideas, improvements, innovations) are valued above hypothetical risks. These kinds of consequences are more likely to occur when information is evenly shared and spread, making it possible for employees to give inputs and make contributions also over the boundaries of their own specific roles and areas of responsibility. Of course, communication overload can cause problems and needs to be avoided. Consequently, the above description should be understood as an illustration of an attitude towards information sharing in trust-laden relations rather than a guide for communication.

Trust enables sharing of responsibilities and duties in situations where some flexibility that exceeds the normal is needed. Although the members of a GSD teams usually have their own roles and known areas of responsibilities, sometimes in a project there will be situations where ad-hoc decision making and quick reacting in necessary. This may be required, for example, in situations where an extensive analysis of a problem, a solution design, or solving some major code problem is called for.

In these kinds of situations, it is beneficial if the responsibility for handling the issue can be given quickly to someone who maybe is not the area expert but has the motivation to participate in solving it. It is impossible for employees to extend their competence areas if they never need to go even
near the boundaries of their current competences. This of course requires not only trust, but also willingness to make decisions based on trust. When people, from time to time, need to work outside of their comfort-zone, they not only strengthen their own skills and competences but also, from the company’s point of view, their capabilities and expertise will be more extensively utilized. Thus, in our view, team level trust and sharing of responsibilities is one of the factors that also powers and enables organizational renewal, learning and innovation.

**Trust is far more easy to lose than get back, and it is most invisible when it is at its highest. Best way to build trust is not to lose it.**

### 4.7 No trust — so what?

What if there is no trust towards some individual or group of people, a developer or project manager or a sub-group of a development team of a GSD project? Based on the understanding of the meaning of trust, the logical consequence of distrust is that the interests, which are assumed to be threatened by the opportunistic behavior by the party not enjoying full trust, need to be protected and risks of loss minimized. When distrust is felt towards an individual(s), the potential trustor gets mentally ready for non-optimal outcomes or behavior by the person trusted.

If distrust is mutual, both parties have assumptions that their interaction will have some consequences that are generally negative or are about to cause some negative outcomes for them at a more individual level. For GSD teams, as well as for any other teams in which tight collaboration and clear channels of interaction between team members are needed, feelings of distrust are destructive. It is quite self-evident that a company with ambitions for a good performance cannot afford to have teams whose abilities to collaborate are threatened by distrust.

Distrust (and of course also trust) is a concept which describes certain feelings towards a person or a group of people. It is a certain way of thinking about an individual in certain situations, and thus it doesn’t necessarily have to be in accord with the reality. It is very possible that although from the trustee’s viewpoint there is no clear reason for the feeling of distrust, or trust, these feelings can occur and be totally genuine.

Following this, from the point of view of GSD teams and other work groups, the first and, according to our view, the most severe consequence of distrust is the mental stress it causes for people who should collaborate
effectively. The feeling that the other party might behave opportunistically or that he/she might not act in a manner or with an attitude conducive to the common goals can be as harmful for the project as real conflicts.

There is a risk that because of these concerns the work tasks or responsibilities are, either consciously or unconsciously, shared unevenly or some other way, which makes the distrust tangible. On the organizational level this means that if the employees are not trusted, their competences, skills and knowledge won’t be fully utilized for the project and the company.

Figure 4.1 describes the self-reinforcing cycle of distrust, in which the team level distrust towards some employees will probably affect negatively the performance of these employees, for example by lowering their commitment towards the project and their general working morale, which as a consequence can make the distrust towards them seem more justified.

Naturally, in a project where there is a considerable need for those team characteristics for which trust works as the essential enabler, lack of trust causes more severe drawbacks. Team characteristics requiring more trust include open and active communication, joint problem solving, high involvement and dedication of all team members, sharing of responsibilities, and extra-role behavior. By extra-role behavior we mean the general willingness to contribute and take responsibilities in an a project in addition to the own role-based tasks and responsibilities.

Often in distributed projects, work and responsibilities are systematically shared between distant units in a very structured way, and interactions and workflow between these units intentionally tries to follow a prescribed formula. This happens, for example, when testing, maintenance or simple bug-
fixing are done in a location different from where the project management, client interface management, and core design and development takes place. It is possible that in such projects trust (and lack of it) may not play such a significant role, because the interaction between the people from different units is not expected to be as dynamic as in projects where more thorough involvement is expected from all the employees of the project. The normal projects goals can be achieved, but it is improbable that some performance would exceed the ”minimum requirements” in that kind of atmosphere.

As a consequence, if GSD teams need to collaborate in a distrustful atmosphere or cope with lack of trust towards some of the group members, the costs will be higher than in teams not suffering from lack of trust. The more collegial and committing the attitude expected from all GSD team’s members is, the stronger the ”handbrake” effect will probably be, from the project performance point of view, if low levels of trust get to affect the team dynamics. Exactly how much higher the monetary costs will be — resulting from, for example, lower quality, required rework, lower morale and motivation and lower communication activity — it is impossible to estimate, as it depends much on the unique nature of the project in question.

Case example:
The next quotations illustrate a project manager with feelings of distrust towards the developers at on-site, without having any obvious reasons for his feelings.

Q: Do you have some examples or evidence to support your feeling that nothing will happen unless there is a possibility to monitor the progress, some real experiences where one is justified in believing that nothing will happen, or is it more like just a feeling?

A: I think it is more like a feeling, I can’t find any clear example right now. Maybe it is because we don’t have any methods for following the tasks and hours of the remote team, which is a clear deficiency in our process. But I don’t have any proof for it, that it would have happened sometime, that no one wouldn’t have done anything there... but one time when I called our developer down there in the afternoon, he was heading to the beach. He said he would continue working in the evening, but I don’t know whether he did or not...
4.8 How to build trust?

"Trust building" is a phrase that is used to describe the content and purpose of sessions where a group of people not knowing each other but having an expectation of future collaboration — as is the case with the GSD project kick-off — meet each other maybe for the first time and start to create a solid ground for collaboration. The first step usually is getting to know a bit about the person(s) you are supposed to work with in the near future. If the company has succeeded in recruiting, these people have the needed professional competencies and also the social skills required from a nice colleague. So is there now a strong trust that helps the team get over all the problems waiting for it? No, probably not yet.

Project kick-off sessions can be very fruitful, and it is a good rule of thumb to try organizing one in the beginning of a project. According to our interviews, these sessions were usually liked, and in the best of the cases served both for social bonding and crucial knowledge exchange purposes. Hopefully, after these sessions, the team members will have positive overall expectations towards the project and their colleagues. But the formation of trust between individuals and groups needs time to grow, and that takes place in the everyday workflow of the project in question. Social relations in the team, including mutual trust, respect, shared norms, commitment to work, and process discipline are based on the information which is continuously carried and delivered along with the interaction in the team.

It is a false belief that by only having a kick-off session the difficult "team building" issues could be taken care of, and these would then need no further attention. In addition to kick-off sessions, good social relations in the team should always be maintained and enforced. Maybe "team building" should be understood more as an attitude towards teamwork than as separate activities.

"Team building" should be understood as an attitude rather than a separate activity.

Employees’ interactions obviously cannot be "managed" to support trust development. A manager of a GSD project who is sensible for team dynamics tries to be aware of the level of trust in his/her team, and tries to enforce the conditions that could strengthen that trust, for example, by enforcing connections and open communication in the team. According to an extensive amount of research, physical meetings are the best way to enforce
trust creation in a team, but in addition to project-kick off sessions, informal work meetings related to the progress of the project itself would probably provide a more efficient context for the creation of trust in a GSD team. In these meetings people get better opportunities to display their competences, to make their way of thinking, working and contributions more visible, and to strengthen their personal roles in the project. These are some of the main factors that are likely to affect the considerations about people’s trustworthiness.

The project manager is one of the key players in managing trust on both sides, and physical meetings are the best situations for these purposes. Probably the managers and supervisors who are respected by the team are influential in providing the team members with those ”good reasons” that may help them have trust on the other team members.

If issues related to low level of trust are acknowledged in a GSD team as problems, that should not be accepted as normal ”status quo” and left without further attention. Probably the best and only way to start fixing the issue is to collect the parties physically together and try to open up a constructive discussion on the issue between the parties. The one who recognizes the problem should be active in making the initiative because it is possible, although unlikely, that the other party is not aware at all about the problem. Probably both parties have at some level recognized the state of affairs, so getting the issue solved would be welcomed for both. A good starting point in these kinds of discussions is where both parties explain the situation from their own viewpoints and make clear why they personally have felt distrust or lack trust towards the other(s). With an open and constructive discussion, it is possible and desirable that the parties get to understand the true reasons for each other’s behavior, which hopefully provides them with new ”material” to be used in the evaluation of each others’ behavior. The most optimal outcome of this reevaluation would of course be the realization that actually there is no rational reason to be concerned about the sincerity of the other.
CHAPTER 4. TRUST AND DISTRUST IN GSD PROJECTS

Case example:

One example of a misinterpretation of the reasons for a project manager’s behavior that caused developers to question his trust towards them, can be found from one of the MaPIT case projects.

The manager of a GSD project wanted to monitor very closely the activities and progress of some developers in the project, who were visiting the on-site premises and trying to integrate to the project via knowledge exchange activities and some research tasks related to the project. The problem arose when the developers started thinking that distrust towards them was the reason for their close monitoring in the beginning. The project manager explained that the reason for close monitoring wasn’t his distrust, but his need to be fully aware of the skills and level of competences of the persons he had not worked with before and who he met first time in the project now.

Maybe the problems could have been avoided if the project manager had honestly told about his intention to closely follow the developers’ work in the beginning.

Building trust is based on interaction and thus requires involvement from all parties. But just one person is enough to cause the break of trust.

Trust overruns the considerations of possible risks. Showing trust strengthens the initial trust.

References


CHAPTER 4. TRUST AND DISTRUST IN GSD PROJECTS

Limerick, Ireland.

Additional readings


Chapter 5

GSD Projects and Organizational Change

Arttu Piri

5.1 Introduction

In our analysis of problems related to group relations in GSD projects\textsuperscript{1}, some of the most important problems had to do with the concerns and even fears that the employees in GSD projects experienced when their previously collocated projects were reorganized and distributed globally.

Our analysis revealed that an introduction of intra-organizational globally distributed collaboration with poorly communicated motivation and goals can cause severe fear and concern about the employees’ own future within the organization at the on-site location. According to our analysis, this fear was often rooted in the way how the change from collocated development to globally distributed collaboration was introduced and communicated at the on-site location by the senior management. Our data indicates that unclear and vague explanations about the core purpose of the transition to GSD, as well as about the long term strategy for GSD utilization made some employees at the onsite location to suspect that the change may cause a threat to their work in general, as well as to the stability of their current job descriptions. Often the change from collocated development seemed poorly motivated. It was implicitly based on the simplistic idea of implementing GSD mainly because of the possibilities it provided in cost-savings. These could be achieved by distributing development tasks to low-cost countries and making changes in the roles and responsibilities of employees at the onsite location. We could call this kind of approach to GSD as ”organizational

\textsuperscript{1}Piri et al. (2009)
Case example:

The next quotation illustrates the distrust towards off-site team members and the concerns related to one’s own professional future in a project.

A: So I think the strategy that the company needs to have is actually to communicate about the effect of having an outsourcing organization: what is it, is it really to produce more software or is it to produce software more cheaply? If it’s to produce more software, if it’s to expand the organization, then you can see that your job on-site is safe. But if it’s a strategy to move maintenance tasks and reduce the responsibilities in Scandinavia, then it’s harder to actually have trust in the persons that you are going to work together with.

As a consequence of the fear that the project members at the on-site locations felt, they openly exhibited self-protective behavior. For example, they tried to protect the old organization of work by purposefully making it difficult for the colleagues at the off-site locations to contribute to the project. Intentional neglect to share or produce information needed by the colleagues at the off-site location, for example by refusing to make proper documentation for the implemented functionality, are examples of this kind of behavior.

In addition to the fear related to the new intra-organizational global collaboration, another problem the projects suffered from was salient distrust between the employees at the on-site and off-site locations. In our interviews, employees at the on-site locations demonstrated both implicit and explicit distrust towards their colleagues at the off-site locations.

According to our analysis, the distrust that emerged between project personnel at the on-site and off-site locations seemed to be strongly rooted in the fear and personal concerns, which were felt by the people at the on-site location and related to poorly introduced global distribution of the work. The fear developed as a reaction to the senior management’s decisions to reorganize the work and utilize the resources which the companies had in their development centers in lower cost countries. Figure 5.1 illustrates the relationships between fear and distrust in the analyzed projects.

As a reaction to the management’s decision (1) to utilize internal global resources, fear and concerns toward one’s own professional future (2) in the company was developed mainly among the employees at the on-site location. We think that these concerns were due to the "opportunistic" approach.
towards globally distributed software development shown by management. This opportunism was seen in the implementation of global distribution, in which the strategy for GSD wasn’t clearly communicated, and which was only vaguely referred to ”cost reductions” that could be reached via distribution. Thus, there was no proper motivation for the employees at the on-site location to adapt with the change in the work organization.

Despite the fact that it was the management whose ”opportunism” caused the fear, it was the personnel at the off-site location whom the distrust was directed to (3). Thus, it seems that the off-site employees had to suffer from distrust that, at least partly, seemed irrational. The concerns related to the future of the project and the fear related to one’s own role in the project also escalated among the employees in the off-site location when they realized what kind of stand their colleagues at the on-site had taken towards collaborating with them (4).

### 5.2 Managing change from collocated to distributed teamwork

Based on the analysis and findings of this study, we suggest a three-phase model to help organizations in preventing the development of fears and concerns related to the change from collocated development to distributed collaboration. The model will also help organizations to prevent irrational dis-
## Table 5.1: Phases for initializing intra-organizational distributed collaboration in GSD

<table>
<thead>
<tr>
<th>Phases</th>
<th>Actions</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Motivating</td>
<td>- communicating the organizational strategy related to GSD</td>
<td>- creation of internal motivation for the change</td>
</tr>
<tr>
<td></td>
<td>- deciding the appropriate collaboration structure</td>
<td>- prevention of irrational concerns</td>
</tr>
<tr>
<td></td>
<td>- clarifying the consequences for individuals</td>
<td></td>
</tr>
<tr>
<td>2: Adaptation</td>
<td>- organizational socialization and integration</td>
<td>- familiarizing with new colleagues</td>
</tr>
<tr>
<td></td>
<td>- building mutual respect and trust</td>
<td>- enabling realistic expectations</td>
</tr>
<tr>
<td></td>
<td>- agreeing about the norms of work, building discipline towards the process</td>
<td>- ground for mutual flexibility</td>
</tr>
<tr>
<td></td>
<td>- knowledge transfer</td>
<td>- efficient and appropriate communication</td>
</tr>
<tr>
<td></td>
<td>- face-to-face interaction</td>
<td></td>
</tr>
<tr>
<td>3: Reorganization</td>
<td>- starting the new, agreed way of distributed collaboration</td>
<td>- flexible collaboration</td>
</tr>
<tr>
<td></td>
<td>- changes in roles and responsibilities</td>
<td>- higher tolerance of problems and disappointments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- higher job satisfaction</td>
</tr>
</tbody>
</table>

Trust between people at on-site and off-site locations. This model is illustrated in Table 5.1.
CHAPTER 5. GSD PROJECTS AND ORGANIZATIONAL CHANGE

In the first phase, it is crucial that the people who are mostly influenced by the change have positive or at least a neutral attitude toward collaboration with distributed colleagues. This can be achieved by open and sincere managerial communication of the way how the company is trying to utilize global sourcing in a given project. It is important to involve the employees at the project management level in the process of analyzing whether global sourcing could be beneficial for the given project, and if so, how the collaboration between parties should be structured to enable optimal outcomes. By explaining the possible change and communicating the personal consequences for all the employees who would be influenced by it and thereby building internal motivation to support the change process, it is possible to prevent irrational concerns and false interpretations of the anticipated outcomes among the employees. Naturally, if the fears and concerns are rational, and if the global sourcing will have a negative effect on the on-site employees personally, their unwillingness for collaboration with their colleagues at the off-site location is cognitively motivated and understandable.

In the second phase, it is important that mutual adaptation to the new social and organizational structure is created. In this phase, the ground for flexible and uncomplicated collaboration will be created. At the core of this is the organizational socialization, by which the employees working at the off-site location are integrated in the project team. Socialization takes place in everyday interactions between the employees, but most efficiently in physical meetings where both tacit and explicit knowledge is shared and where norms for collaboration and social relation are created and maintained.

Efficient socialization prevents issues from escalating to problems between people, and helps in creating a mutually proactive attitude toward solving problems faced during the project. Although knowledge sharing is a crucial facilitating element for collaboration and thus needs to be kept active throughout the whole project, in this phase it is important to make sure that an appropriate “culture” for knowledge sharing is adopted by all project employees. In this phase, face-to-face interactions and concise communication between people facilitate the adaptation process. As a result, the ground for setting realistic expectations towards the behavior and contributions of the colleagues will be created, which reduces the possibility of encountering surprising problems and disappointments in the project.

According to our view, changing the work processes is especially risky before the adaptation phase has strengthened the social structure of the distributed project. The needed level of organizational, psychological, and competence-based flexibility, which enables a smooth transition from the old to the new structure, is created in the adaptation phase. As time goes by, the new project organization and the collaboration structures between the
CHAPTER 5. GSD PROJECTS AND ORGANIZATIONAL CHANGE

distributed teams will be normalized.

Depending very much on the individuals involved, the nature of the project, earlier experiences, and in general on the unique project-contextual factors, creation of a solid base for reorganizations may not happen very fast, and some extra resources are needed to facilitate and coordinate the change. Many experiences from troubled GSD projects suggest that there is no shortcut by which the promises of GSD could be easily reached.

To be able to enjoy of the GSD paybacks, cost savings among them, companies need to be willing to invest more to deal with HR issues, in organizational learning and in general project planning than in collocated projects.
CHAPTER 5. GSD PROJECTS AND ORGANIZATIONAL CHANGE

References


Additional readings


Chapter 6

Structuring the Teamwork for GSD Projects

Arttu Piri

6.1 Introduction

In addition to problems related to trust, another issue that seems to cause problems in GSD projects at the level of group relations is the low level of ”teammness” in the projects. By this we mean that, according to our interviews of project employees, team leaders, and project managers, a certain kind of sense of belonging together and sharing a common interest in context of the software project is emphasized by these groups. It seems that the members of GSD projects are generally willing to see the projects more as group efforts than just as a work where the contributions of separate individuals are put together.

In our view, this kind of approach for making software is the most natural one, as the software projects generally speaking require highly dynamic interacting between the team members. Software projects seem to be based on knowledge exchange, problem solving and knowledge creation, between the often specialized software professionals who are related to the software product being built.

One obvious ”proof” of the fundamentally collaborative nature of software development is the fact that most industrial software development projects are established around a team, a group of people with competences that fit to project needs. Team-based organization is not the only possible one for organizing software development. But software teams exist because they are needed.

However, according to an analysis of MaPIT case projects, the GSD
projects seemed to suffer from unclear team structures. In general it seemed to be a bit unclear what were the roles of the teams in these projects, who were supposed to belong to which team and who weren’t, and how the teams should work together. From the viewpoints of both onsite and offsite employees of those projects, the expectations towards the "teamness” were rarely satisfied in these projects.

One reason for this is obvious; in GSD projects the employees are supposed to form a team and work as a team in conditions where usually only some members of the whole group are able to work close to each other in the same physical location. In our case projects, the development project teams were often distributed between two separate locations: the onsite teams (usually responsible for project management, design & planning, client management) at Scandinavian countries and the offsite teams (usually providing resources for development, testing and bug-fixing) in the company’s internal resourcing centers in Eastern Europe or India.

Already the above simple description contains the main issue from the point of view of team formation in GSD projects. Is the distributed development project formed around a single development team, consisting of employees located both at the onsite and the offsite, or are there two separate teams at the onsite and the offsite? Should there be one team or two (or more) teams in these GSD projects? The distinction may seem trivial and merely theoretical, but probably it is not.

A critical view towards the trend of global distribution, if such existed, would probably claim that in globally distributed software projects attempts are made to maintain team driven software development in conditions lacking some of the fundamental elements for genuine team formation, above all the physical proximity. Based on the amount of problems in GSD projects in general, these kinds of critics would make quite a strong empirical case; nevertheless, there are no signs of abandonment of team driven software development in GSD projects. Actually the opposite is true, as agile methods and especially Scrum, which relies heavily on the capabilities of agile team, are gaining ground also in GSD.

So only option left for companies trying to improve their performance in software development is to enrich understanding of how conditions of global distribution may affect the team dynamics in software development, what are the factors that get the best out of the team, and how these two could be combined in GSD context.
6.2 Team integration and team cohesion

Team’s can be described and analyzed with concepts like team cohesion and integration to the team. By team integration we here mean the process by which new employees of GSD projects will be integrated to become members of their development teams. Team integration can also be used to describe the quality or the level of integration, i.e. how well the employees are integrated to the team.

The more cohesive a team is the more strongly the members of it will stick together while pursuing the team goals. As an example, a team or work group having a high voluntary turnover is not very cohesive when compared, for example, with a team which has remained with the same composition of individuals for years.

In the previous chapter it was described how software development is very much a collaborative effort of a software development team, a group of professionals who are supposed to be able to take care of their responsibilities cooperatively in the project. According to the interviews conducted in the MaPiT project, the resource planning made in the GSD project ramp-up phase often included selection and recruitment of several employees. At the onsite, employees were usually allocated to GSD projects according to the availability of internal resources with suitable competences, and new recruitments from outside the company were rarely made.

But when it comes to employees at offsites, it was very common that they were recruited from outside the company and that the projects studied were their first with the company. In addition, compared to project employees at the onsite, the employees at the offsite often had fairly limited work experience in software development.

So from the integration point of view, in many of the projects studied there seemed to be at least two topical issues:

- integrating new external recruits, located at the offsite, to the company; familiarizing them with company policies and culture
- integrating the internal and external recruits to the project at a team level; familiarizing them with project policies, practices and culture, and fostering a "teaming” process to create a team(s) structure suitable for the project

While the first of these issued is related to organizational practices on a level higher than the focus of this section, in the following part we focus on discussing team level integration and optional ways to structure team collaboration in GSD projects.

We limit our analysis to focus on two basic approaches to structure team collaboration in GSD projects: the ”one-team model” and the ”separate
teams model”. In real world, of course, GSD projects probably can’t follow purely either one of these rather simplistic models. Since the projects and groups are always unique and form their own unique structures, these models should again be understood as illustrations of two different approaches to GSD team formation, rather than as exclusive options to choose from.

6.3 One-team model in GSD projects

The basic idea of the ”one-team” -model is illustrated in Figure 6.1. The basic idea of this approach is that all members of the distributed development team in a GSD project should form one team or work group and work as such, without making either a formal or informal distinction based on team members’ physical location or some other factor. With this approach, there shouldn’t be any ”onsite team” or ”offsite team” or even ”onsite team members” or ”offsite team members” but just ”our team”. There shouldn’t be ”us” and ”them”, but only ”us”. So in the one-team model the idea of high level integration is inbuilt. Also high cohesion, willingness to stick together, and collegial trust are the nominators in group work based on the one-team model.

The one-team approach in GSD projects should emphasize the importance of all team members’ active participation in problem solving and their contribution to project progress regardless of their specific role in the project. Probably in every GSD team there is some kind of role structure based on professional specialization or personal competences of employees. But the problems faced in the project should be met collectively, and not owned only by those who happen to be closest through their role.

From the point of view of communication and interaction, GSD projects
set a serious challenge by restricting the possibilities for face-to-face interaction only to situations in which the whole team has managed to get in the same location for a period of time. Without the possibilities for continuous face-to-face communication, the employees need to maintain intensive communication and maintenance of social relations with the communication tools available (see Chapter 10 on communication tools).

If the goal is to foster the formation of the one-team model in GSD projects, the role of the factors creating motivation and commitment towards the shared objectives should be carefully taken into account. If the team members have very different expectations for personal consequences resulting from, for example, completely failed or highly successful project, it may be difficult to establish a truly joint effort of the project. In the GSD context, significant differences in wages between employees at the onsite and at the offsite may be a disruptive element for the formation of one-team atmosphere; the consequences are probably more severe if only a part of the team feels getting rewarded for a high quality work and project success.

The most crucial factor affecting the success of the one-team model is however the level of acceptance and support for it from the team members. If the team members, wherever they are physically located, will not “buy” the basic idea of the one-team model themselves, there are no chances for success in creating a tight team which could provide the benefits of what DeMarco & Lister\(^1\) describes as the results of a ”jelled team”.

### 6.4 Separate teams model in GSD projects

The idea of the separate teams model is illustrated by Figure 6.2. The key idea in the separate teams model is that instead of trying to create a shared identity and feeling of belonging to the same team the employees both at the onsite and at the offsite are, on purpose, encouraged to create and maintain their own teams. In the separate teams model, the physical separation between the employees does not negatively affect team formation, because both the onsite and the offsite employees are considered to form their own teams. From the collaboration point of view in GSD, the question is then of collaboration between the teams and not within one team.

The separate teams model is probably most suitable for situations where interaction between the onsite and the offsite is not expected to be very intensive at the employee level, and when employees at the onsite and at the offsite clearly have different roles and different opportunities to affect the project success in general. The separate teams model still requires that both

---

\(^1\)DeMarco & Lister (1997)
teams have a clearly defined purpose and role in the project, as well as clear boundaries (i.e. who belongs to the team and who doesn’t), providing a solid base for employees’ identification and integration to the team.

With the separate teams model, the communication between the employees in distinct teams is probably less intensive and dynamic than with the one-team model. The communication in the separate teams model is probably concentrated to happen within the teams, which is natural if there is not much overlap in the teams’ roles in the project. Probably both the onsite and the offsite teams also have their own team leaders who are working as ”links” between the teams and who are responsible for coordinating the work and tasks in the teams.

With the separate teams model both the onsite and the offsite teams should have an efficient structure for generating motivation and commitment towards team-specific goals in the project. But as described in the previous chapter, in GSD projects it may be the case that the onsite and offsite teams do not share the same incentives to encourage the teams to aim for high quality deliveries. There is also no reason to believe that the employees at the offsite need less motivation related to their job than their colleagues at the onsite, so both the onsite and the offsite teams should have a solid and effective incentive structure fitting to their working conditions.

When collaboration is expected to take place, some sort of a contract needs to be created to tie the collaborating parties together. Usually formal and psychological contracts are put into different categories. It is very
probable that in the separate teams model there is a formal contract between the onsite and the offsite units, clearly specifying the terms of collaboration from the point of view of monetary costs, management & supervision, and responsibilities & obligations from points of view of the collaborating parties. Thus the key driver in collaboration is the formal contract, in the GSD context an agreement where the offsite unit, the "seller", promises to provide a certain amount of resources for the usage of the onsite, the "buyer".

On the other hand, the psychological contract can be described as an internal obligation that individuals in a group have towards each other and that motivates them to work to achieve their shared goals and objectives. The one-team type of collaboration cannot be achieved merely by formal contracts. From an individual's point of view, the formal and psychological contracts do not necessarily need exclude each other. But from the point of view of one-team type of collaboration in GSD projects, there needs to be also psychological contract between the team members supporting their shared commitment towards the project and driving them to dynamic, flexible and solution-oriented collaboration.

### 6.5 Experiences from MaPIT case projects

According to our analysis of MaPIT case projects, at the "idea level" most of the teams tried to achieve a one-team model, at least at some extent. The interviews of project managers and team leaders indicate that an approach where "all our team members belong to the same team" was seen as some kind of a norm, towards which all GSD projects should try to aspire.

But at same time it seems that the projects had many practices that were disruptive for the "one-team" vision. Typically employees both at the onsite and the offsite described the teamwork in a way that implied more the separate team model than the one-team model. Only in few projects there were employees who said that they identify themselves as members of a group consisting of employees of both the onsite and the offsite, to quote "in our project all the employees belong to the same team".

Despite the "normative" attitude towards the "one-team" ideology, there seemed to be a clear separation between employees located at the onsite and those at the offsite when it comes to their roles, responsibilities and task division, access to knowledge resources, involvement in planning and decision making and communication practices. Typically the employees at the offsite took care of software testing, participated in the development or handled tasks related to bug fixing and maintenance. They basically never got an opportunity to participate in product planning and design, architecture design, or project management. Also the language usage, referring to the
"offsite resources", "test team", or "low-cost employees", constructed and reinforced the distinction between the physically separated employees.

Another issue not very supportive for the one-team model was the "inertia" towards the suggestions and ideas of the offsite employees, usually expressed by the project and team leaders and employees responsible for product design at the onsite. The remote members of the teams reported that their onsite colleagues often seemed not to take their suggestions very seriously. According to our interviews, there were basically two different ways of reasoning about this inertia, and both of those ways are problematic from the point of view of the one-team model. The first possibility is that ideas and suggestions originating from remote employees were not taken into consideration just because of their origin. The second possibility, and probably the more accurate one in the projects studied, is that these suggestions and ideas weren’t feasible because some important detail or restriction related, for example, to the functionality of the product weren’t taken into account in these suggestions. According to the offsite employees, in many cases the main reason for this was that the offsite employees weren’t aware of some important issues that made their suggestions unfeasible. The question then is: why didn’t the remote employees have an access to knowledge resources that could provide an equal opportunity with the onsite employees to contribute to the project by improvement suggestions and ideas?

Insufficient levels of project domain knowledge at the offsite was a common problem in the projects, and only in few projects the issue of providing all team members with appropriate knowledge of the product and the domain was handled systematically. Despite that in some projects class-room training and knowledge exchange workshops were organized, the knowledge transfer and exchange processes were usually based on ad-hoc informing sessions rather than keeping the remote employees constantly in the loop of knowledge.

In the studied cases the project ownership was always with the Scandinavian (onsite) units of companies. The project lead was located at the onsite, and usually the development project manager was supervising the development employees both at the onsite and at the offsite. But in many cases, employees located at the offsite had an additional leader of their own, often called the "offsite team leader", who usually were supposed to represent the offsite group and to keep the wheels rolling at the offsite, and together with the team leader / project manager at the onsite, coordinate development work in the project. Having own team leaders both at the onsite and at the offsite can be seen as a clear indicator of the "separate teams model" and contradicting with the ideas of the "one-team model". If all the employees are to form a joint team and have an equal status in the group, why should
there be a separate supervisor for the offsite employees? Part of the explanation for this is probably rooted in the logic of how economical risks and possible profits related to GSD projects were shared between the onsite and offsite units in the companies studied. In our understanding, based on the case interviews, the standard way of action was that the offsite unit would just provide resources for the project according to its needs and charge the project by the number of hours or level of work-time dedication that the offsite employees had provided for the project. And, of course, it is the buyer who has the responsibility and right to utilize the purchased ”goods” in the best possible way, according to his own will.

Because the internal offsite units were utilized mainly as resourcing centers offering lower costs workforce for projects owned by some other units for example in Scandinavia, it meant in practice that the responsibility for allocating tasks for offsite employees and ”keeping them busy” was entirely with the onsite. This strategy towards globally distributed software development was clearly represented by the way how tasks and responsibilities were allocated in the projects and by the fact that both the ownership of economical risks and profits of the projects were wholly at the onsite. Thereby it not difficult to understand why the project managers and team leaders at the onsite often described their concerns and suspicions related to the quality of the deliveries, level of project commitment, level of dedication, discipline towards processes and deadlines, and even the level of trustworthiness of the employees at the offsite.

Roughly put, it seemed that it didn’t make much difference to the offsite whether the projects they had provided resources for ended up being successful and profitable or not. It seemed that the offsite units often didn’t have much stakes in the game aiming for project success either from the point of view of a single employee or the offsite unit generally2.

So there seemed to be quite many discrepancies in the projects between the hopes and expectations towards the nature of team level collaboration in GSD projects and the real-world structures providing the context for collaboration. It seemed that both at the onsite and at the offsite there was willingness towards the one-team model and a quite strong understanding that the project would benefit if the employees could work together more as one team. The idea of making the employees at the offsite more thoroughly involved with the projects was widely, although mostly implicitly, appreciated. However, it seems that the structures of collaboration in the MaPIT case projects did not provide very good opportunities for practical implementations of the one-team model.

2This notion is mostly related to product development or projects having an external customer, and not to company internal development projects.
In our understanding, many of the problems encountered in these projects and related to managing the distribution of software development activities (i.e. structuring & coordinating the task allocation, quality of the deliveries, discipline towards processes, effort estimation & keeping the deadlines) can be at least partly rooted on the conflict between the expectations towards the performance of GSD teams, which are mainly based on earlier experiences of performances of collocated software development teams, and the real-world conditions, which provide the context for team-level collaboration in GSD projects. In the GSD context, the expectations towards team performance need to be estimated and adjusted in relation to conditions of collaboration provided in a GSD project.

If the conditions in the project are supportive for the one-team type of collaboration and a psychological contract between the employees, the expectations towards the team collaboration and team performance can be set accordingly. If it seems, after an honest reflection on the available options for structuring a GSD project and the most realistic evaluations of the possible collaboration models, that the separate teams type collaboration is more suitable for a given project, the expectations toward the collaboration should, again, be set accordingly.

In the tables 6.1, 6.2 and 6.3 we have collected some basic factors, both supportive and disruptive, for the formation of one-team and separate teams models of collaboration in GSD projects.

In addition, we provide some ”self-diagnostic” questions mainly for the benefit of GSD project managers, hopefully helping them to sort out the basic characteristics of their projects (current or forthcoming) and to help them in directing the project towards a model of collaboration which could provide the best prerequisites for its success.
### Table 6.1: Supportive factors for the one-team -model

<table>
<thead>
<tr>
<th>Supportive factors for the one-team -model</th>
<th>Questions for self-diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Respect towards every team member both as an individual and as a professional</td>
<td><em>How much do you know about the members of your team?</em></td>
</tr>
</tbody>
</table>
| Equal access to knowledge resources | *Does every team member have equal access to all knowledge repositories?*  
|  | *Is the knowledge shared with the team members only functionally, based on their immediate needs?* |
| Physical closure | *How often has your project team met physically?*  
|  | *How often are you going to meet?* |
| Active and unrestricted communication structure | *How often do the team members communicate?*  
|  | *Does everyone take an active part in communication?* |
| Equal openness to ideas and suggestions regardless of their origin | *Do you value the ideas and suggestions, regardless where they originate from?* |
| Making all the supportive practices clear to all team members; supporting everyone in personally applying and improving them | *Have your team discussed about how to form and maintain a "one-team" atmosphere?* |
| Mutual trust | *Do you trust your team members located both at onsite and offsite*  
|  | *How do you show your trust?* |
| Employees’ own will to maintain the one-team model | *Have your team members genuinely bought the idea of the one-team model?* |
| Psychological contract | *Do the team members have a strong internal motivation for project success?* |
Table 6.2: Disruptive factors for the one-team model

<table>
<thead>
<tr>
<th>Disruptive factors for the one-team model</th>
<th>Questions for self-diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differences in incentives and reward structures for the individuals</td>
<td>Do all team members share the same motivation to aim for project success?</td>
</tr>
<tr>
<td>Differentiation in role structure and task division based on importance, &quot;easiness&quot; or needed competence level</td>
<td>Is the offsite team only responsible of the more &quot;simple&quot; tasks?</td>
</tr>
<tr>
<td></td>
<td>Are the most important decisions made at onsite?</td>
</tr>
<tr>
<td>Uneven distribution of knowledge and access to knowledge resources</td>
<td>Are there restrictions for access to project relevant information?</td>
</tr>
<tr>
<td>Selective openness to ideas based on their origin</td>
<td>Do you think that every team member is valuable in creating new ideas?</td>
</tr>
<tr>
<td>Communication practices implying exclusion</td>
<td>Are there meetings which are only for onsite/offsite team members?</td>
</tr>
<tr>
<td>Lack of appropriate technical support to cope with physical dispersion</td>
<td>Are you happy with the availability and usage of communication tools in the project?</td>
</tr>
<tr>
<td>Physical separation</td>
<td>Is the team most of the time separated from each other?</td>
</tr>
<tr>
<td>All the team members don’t have the same supervisor</td>
<td>Is there a separate offsite team leader?</td>
</tr>
<tr>
<td>Distrust</td>
<td>Do you have equal trust towards the employees at onsite and offsite?</td>
</tr>
<tr>
<td>Vocabulary implying separation</td>
<td>Are you used to talk about &quot;onsite team&quot; and &quot;offsite teams&quot;, or &quot;test team&quot; and &quot;design team&quot; etc.?</td>
</tr>
</tbody>
</table>
### Table 6.3: Supportive and disruptive factors for the separate teams –model

<table>
<thead>
<tr>
<th>Supportive factors for the separate teams -model</th>
<th>Questions for self-diagnosis</th>
<th>Disruptive factors for the separate teams -model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intended and clearly motivated division of employees to separate teams</td>
<td>Is it clear who belongs to which team?</td>
<td>Unclear team boundaries</td>
</tr>
<tr>
<td>There are separate supervisors / team leaders for a subgroups of the project team</td>
<td>Are there any ambiguous supervisory responsibilities?</td>
<td>Unclear hierarchy between onsite and offsite team leaders</td>
</tr>
<tr>
<td>Communication is concentrated within a one team</td>
<td>Is it easy for employees to reach the parties that are mostly needed in daily work?</td>
<td>Constant communication and cooperation between onsite and offsite employees required</td>
</tr>
<tr>
<td>Physical closeness within a team</td>
<td>Are the employees of the team(s) located close to each other?</td>
<td>Even at both sites the employees are not working the same premises</td>
</tr>
<tr>
<td>Clear division of roles and tasks between onsite and offsite</td>
<td>Is there a lot of tasks where connections between onsite and offsite are required?</td>
<td>Offsite team is utilized only for supporting the onsite team</td>
</tr>
<tr>
<td>Site-specific compensations structure</td>
<td>Is the compensation structure as motivating both for onsite and offsite employees?</td>
<td>Compensation structure is not encouraging all employees to create high-quality outcomes.</td>
</tr>
<tr>
<td>Collaboration is based on formal contract</td>
<td>Is there a formal agreement for offsite to provide resources for the project?</td>
<td>Collaboration is based on shared psychological contract</td>
</tr>
<tr>
<td></td>
<td>Is the &quot;project ownership&quot; completely in the hands of the onsite?</td>
<td></td>
</tr>
</tbody>
</table>

91
Key questions for defining the team structure in GSD projects:

- What is the main motivator for global distribution in the given software project?
- What are the short and long term goals to be achieved by global distributing in the given project?
  - Simple cost-reductions, utilization of decentralized resources, proximity with markets/clients
- Does the used contract model support the strategic goals for distributed development in the given project?
- Does the contract model support development of appropriate group relations between employees at onsite and offsite?
  - Do you need to build and develop relations following the one-team model or the separate teams model?
CHAPTER 6. STRUCTURING THE TEAMWORK FOR GSD . . .

References


Additional reading


Chapter 7

Commitment in GSD Projects

Peitsa Hynninen

7.1 Introduction

In this chapter we will discuss the impact of commitment for GSD projects. Commitment has been shown to have impact on organizationally beneficial phenomena ranging from job performance\(^1\) to organizational citizenship behavior, i.e. going beyond the call of duty to help the project and company to succeed.

We also found commitment to be a relevant aspect for GSD in the case projects studied. The case projects were mostly begun at on-site, whereas off-site often only took part later on in the projects. In most cases off-site also had no contact with the customer and so the problems related to lack of commitment appeared there more severely. Decisions, e.g. about what practices to use and what tasks to distribute in the projects, made at on-site could affect the commitment felt towards the project at off-site. An on-site tester in one of our cases described the responsibility they had in making the off-site committed to a project in the following manner:

"They [off-site team members] are committed to work together with us, and do a good job, when we manage to support them. So I feel the co-operation with off-site is depending upon us."

—On-site tester

More often than not, however, the support needed was not readily available. Next, we will discuss how commitment and its antecedents are presented in the literature and how this seemed to fit the situations in our case

\(^1\)Jaros, 1997
projects. We will describe the problems which lack of commitment seemed to inflict on our case projects. Lastly, we will give some insight into how these problems may be avoided.

7.2 Commitment and team work

Commitment is often viewed as consisting of several distinguishable components: affective, continuance and normative commitment\(^2\). Affective commitment refers to the employee’s emotional attachment with the organization or team and is perhaps the form of commitment which is most commonly thought of when talking about commitment in general\(^3\). Continuance commitment on the other hand refers to the perceived costs to the employee caused by leaving the organization or team. Normative commitment refers to the belief that an employee has obligations to the organization or team. The reason why we need different conceptualizations becomes apparent when we investigate their relationship, for example, with job performance. Affective commitment has been found to positively influence job performance whereas continuance commitment has been found to correlate negatively with job performance\(^4\). Affective commitment is generally considered the most important component in regards to performance. This is why we will concentrate on that aspect of commitment in the rest of this chapter.

7.3 Affective commitment

Literature categorizes the possible antecedents for affective commitment into three groups: organizational characteristics, person characteristics and work experience characteristics. For the purpose of this study, organizational characteristics and work experience characteristics are the most relevant and will be discussed next. We will also discuss how affective commitment affects voluntary turnover, i.e. employee leaving the organization out of one’s own decision.

7.3.1 Organizational characteristics

The most important organizational issues regarding affective commitment seem to be perceived fairness of organizational policies, compensation and

---

\(^2\) Allen and Meyer, 1990  
\(^3\) Yeatts and Hyten, 1998: pp. 64  
\(^4\) Meyer et al., 1989
strategic decision making. Konovsky and Cropanzano\textsuperscript{5}, for example, reported higher affective commitment among employees who felt that the organizational explanation given to them about a new policy was adequate for the situation. Distributing projects to cheaper cost countries may cause anxiety at on-site. Good care should be taken about how the reasons for distribution and its effect for on-site employment situation are communicated. Additionally, off-site often has lower wages and they may not be entitled to bonuses if the project succeeds. This may be considered unfair and be a source of discontent, which may result in loss of commitment.

7.3.2 Work experience characteristics

Work experience characteristics is the most studied group of antecedents, and it is here that we also find the strongest and most reliable correlations with affective commitment. The challenge a job presents, degree of autonomy in work and the variety of skills required for the job\textsuperscript{6} all reportedly have a positive influence to commitment. A large number of studies have also shown that role ambiguity (not knowing what is expected from one in work) and role conflict (inconsistent behavior required in work) can lower one’s commitment\textsuperscript{7}. The last aspect in this category which seems to affect commitment is leadership. Consideration and fairness towards subordinates as well as participatory decision making can all positively impact the commitment felt in the team. Making clear who is responsible for what, showing interest in the work done at off-site, and giving off-site some authority, e.g. in effort estimation, may thus be beneficial for commitment.

7.3.3 Voluntary turnover

Affective commitment has also been found to negatively correlate with voluntary turnover, i.e. employee leaving the organization out of one’s own volition. This is suggested to cause certain costs to the organization, for example by means of additional training costs and loss of knowledge. The antecedents found for commitment and willingness to stay in organizations are quite similar. There is one important addition however. If a team member has very good relationships with co-workers he or she may be less willing to leave the company. On the other hand, if those relationships are bad, this may give one more incentive to leave. Co-worker relationships can thus be important in the decisions to leave or stay.

\textsuperscript{5}Konovsky and Cropanzano, 1991
\textsuperscript{6}Steers, 1977
\textsuperscript{7}Mathieu and Zajac, 1990
CHAPTER 7. COMMITMENT IN GSD PROJECTS

7.4 Challenges

In the software projects studied we found two major issues regarding commitment. The first problem lay with the off-site’s seeming lack of commitment to deadlines. This is exemplified in the following quote:

"For example, their schedule promises have usually not been accurate [...] or when something is supposed to be finished. But I don’t know if it is a cultural difference or ... and then it has happened a couple of times that there has been some critical issue and they have said that ‘Yes, I will fix this’ and then nothing has happened."

– On-site consultant

There were at least two important reasons for off-site’s lack of commitment on deadlines. First, in most of our cases, off-site had no contact with the customer, and they did not feel the pressure which the customer put to on-site if the deadlines were not met. Second, as the effort estimation was in most cases done at on-site, off-site felt that they were not obliged to deadlines which were not realistic in the first place.

Another challenge was that in many cases the personnel at off-site was in a constant state of fluctuation. Managers at off-site had hard time keeping their project members in the projects. A large portion of off-site team members either changed to other projects or left the company altogether. Again there are a number of reasons for this kind of behavior. The technology used in some of the projects was old and the off-site team members felt they could not learn anything useful regarding their future careers. The tasks which were distributed to off-site were often regarded as boring: bug fixing and maintenance instead of development and design activities. Lastly, in many of our cases, off-site personnel felt frustrated because they could not take part in effort estimation, although they would be blamed if they could not meet the deadlines they were given.

In many cases, voluntary turnover was a large problem to the off-site project managers. But especially in one case the problem was so aggravated that they had begun to hire persons with no previous coding experience whatsoever because the whole region seemed to be devoid of experienced people.

"Here in off-site, there is no more people. There is not more experienced people. It means that every new man in our team is taking some capacity of other team members."

– Off-site developer
"He’s leaving only to another project [. . .] he will do some web application, and so on. It’s something he wanted to do here. It’s good for him, I think that he’s doing a good thing that he’s leaving ’cos it’s good for his personal growth. [. . .] And from this month, we have a new man here, which will work on his place. This new man doesn’t know anything about programming. And he was hired as developer, and I don’t know what I will do with this.”

– Off-site developer

It is not hard to see the problems which these projects faced. Personnel fluctuation created unnecessary training expenses and severely ate the working time of the few remaining off-site developers. Schedules were slipping further, as no one had time to do the actual development and bug fixing for which off-site had initially been ramped up.

7.5 Solutions

Based on the literature and the cases studied, we will now present possible methods which could help improve the commitment at off-site. An interesting thing to notice is that in one particular case, many of the on-site team members seemed to think that the problem was cultural and thus they were incapable of improving the situation. In the other projects within the same company that we studied, however, the on-site personnel felt that they were in large part responsible for the commitment felt at off-site.

**Prefer distributing new projects.** Distributing on-going projects brings at least three kinds of additional problems compared to distributing new projects. First, the responsibilities in the project have already been divided between the existing project team members, who have taken ownership of these responsibilities. Relinquishing this ownership may create uncertainty and fear, leading to behavior that is disruptive to collaboration between the distributed sites. If the distribution is not communicated properly, on-site personnel may fear that their jobs are in jeopardy. In one of our cases, the on-site personnel also complained that their most interesting tasks (actual development tasks) were distributed to off-site, and they were left with tedious word-coding, i.e. design tasks. This can weaken the commitment felt at on-site. This kind of problematic is discussed more thoroughly in Chapters 4 and 5. Second, if the project is already thin on resources, adding distributed resources may not solve its problems. Off-site personnel often need to be trained by on-site personnel. Additionally, on-site personnel have to spend a lot of time in communicating and answering questions. This may only
make the project less effective. Third, altering the project plan and practices used, to align them with the needs presented by the distributed site, takes further time and resources. All these problems add up and may create a situation where on-site does not have the resources to give off-site the support it needs in order to make it a productive part of the project. By distributing new projects, some of the problems mentioned above may be avoided. Project practices may be planned from the start to be better fitting for distributed projects and schedule and training issues might be considered more thoroughly.

**Arrange training and team building.** When a new off-site team is ramped up for a project, one of the first activities is training. This is also a perfect time to let on-site and off-site get to know each other. Some kinds of team building activities in addition to training are recommended to build up cohesion between the sites and lowering the threshold to communication. Training should be properly planned, and should include hands-on training as well as domain knowledge training. Domain knowledge allows off-site personnel to see the bigger picture, which gives them a chance to take more initiative when problems arise. Broad training also allows off-site to adopt more responsibilities when necessary. off-site should also understand the business logic of the case project. It is a wholly different thing to work for a customer who pays for the project expenses than to make a profitable product with the company’s own expense.

**Arrange frequent visits.** A consistent finding among the more successful cases seemed to be that they spent more resources on frequent visits between the sites. Besides meetings and actual hands-on work together, these visits are a good way to provide information for off-site about the progress of the project. In one of the cases we studied, successful iterations were also rewarded with shared team building sessions to which the off-site team and the visiting on-site members took part. The sessions included, for example, paintball and sauna or eating out. Socializing people across sites seems to lower the threshold of communication and make the off-site personnel feel as a part of the project instead of just a cheap resource.

**Involve off-site to effort estimation.** Giving the off-site team members the possibility to take part in the effort estimation process was reported to increase their commitment to the task deadlines. The initial effort estimation for off-site tasks can, for example, be done at off-site after which it is reviewed at on-site. It may be more useful to give off-site some autonomy over the schedules of the tasks than the prioritization of the tasks itself. In some cases where we saw the exact opposite of this, usually the tasks were done in the wrong order and with slipping schedules.

As a summary, the costs incurred from lack of commitment and volun-
tary turnover can be substantial. In this study, the case projects where lack of commitment seemed to be a problem were struggling with their schedules and quality. In comparison, one of the projects with higher commitment was used as example of successful distribution inside the case company. Table 7.1 summarizes the challenges we identified and possible solutions to the problems involved.

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Possible solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barrier to communication between sites. Especially off-site members not willing to contact on-site when having problems.</td>
<td>Socialize people between sites already at training and ramp-up. Frequently visit other sites and arrange possibilities for working together face-to-face. Reward successful milestones (e.g. iterations or deliveries) with shared team building sessions.</td>
</tr>
<tr>
<td>Constant failure to meet task deadlines.</td>
<td>Involve all team members in effort estimation. Involve off-site with the customer. Make off-site understand the business logic of the project.</td>
</tr>
<tr>
<td>High employee turnover at off-site.</td>
<td>Plan the project from the start to be distributed. Incrementally give off-site additional development and design tasks. Keep both sites informed about the future course of the project.</td>
</tr>
</tbody>
</table>
References and additional reading


Chapter 8

Managing Cultural Differences in GSD Projects

Roberto Evaristo, Ph.D.

Research Fellow, University of Minnesota, & Manager, Knowledge Management Program Office, 3M

8.1 Introduction

It is well known that different countries, regions and many times even cities differ from one another in multiple ways. Some of these differences can be explained by several factors, such as national culture, regional and local affiliations, religion, and other beliefs.

These differences can create performance problems when intercultural teams need to interact to perform a highly coordinated task such as distributed software development. In the remaining of this short summary, a few of the key differences, its effects, and how to manage them to increase team performance will be discussed.

8.2 Description of cultural differences

The management literature typically groups cultural differences around certain characteristics. For instance, Hofstede\(^1\) describes uncertainty avoidance as a national level cultural characteristic, and positions over 40 countries on a scale that goes from 100 (very high uncertainty avoidance) to 0 (very low uncertainty avoidance). Most countries are away from the extremes of the scale. The argument is that, on average, an individual of a country with

\(^1\)Hofstede (1980, others)
very high uncertainty avoidance would want fairly specific and spelled out requirements for a given job, whereas an individual from a low uncertainty avoidance culture would be more comfortable with ambiguity.

It is easy to imagine the impact of such a difference in distributed software development, particularly when the teams involved have members from cultures that are widely different in their approach to uncertainty avoidance. This observation creates many guidelines on how to approach the elicitation and write-up of software requirements, as well as improvements in the process of request for clarifications and check up milestones on project status. Continuous development, in particular, needs to be particularly aware of such differences.

The difficulty starts when one has to consider say, five dimensions suggested by one author\(^2\), seven by another\(^3\) and yet other dimensions developed by multiple researchers.

### 8.3 Redefining culture

A different approach may be to think of culture as a set of expectations that are very likely to direct behavior. Evaristo (2007, p. 203) describes an interesting cultural vignette:

"In Brazil, as a guest leaves a home, it is not acceptable for him or her to actually open the door: the meaning of that is akin to ‘I will never come back.’ Instead, the host is supposed to open the door. In the U.S., on the other hand, the fact that the guest opens the door implies that he or she is freely choosing to leave, instead of being asked to leave. Quite a dilemma may ensue if a Brazilian guest is trying to leave a US house where he or she has been a guest say for dinner. The Brazilian does not want to open the door him or herself, because that would be tantamount to offending the host (in the guest’s mind), whereas the host may be wondering why the Brazilian does not open the door, and does not want to do so because that would offend the guest (in the host’s mind). The impasse generated has the potential to offend at least one of the parties, creating a long lasting feeling of discomfort.”

Expectations of behavior are an integral part of culture and can be a very

\(^2\)Hofstede (1980)
\(^3\)Trompenaars (1997)
CHAPTER 8. MANAGING CULTURAL DIFFERENCES IN GSD . . .

effective way of understanding and addressing cultural aspects in a task specific situation. The problem in the situation described above is the surfacing of those expectations which are not obvious to the stakeholders prior to the event described.

Evaristo (2007, p. 204) offers another vignette from a distributed development software project between Brazilian and US programmers:

”A Brazilian programmer was quite baffled by the reaction of his U.S. counterpart to what he considered to be very polite system requirements clarification questions: coldness, progressive distancing and decreased knowledge sharing over time, quite the opposite hoped for. The problem seemed to be in the words used by the Brazilian programmer: ‘I have a doubt regarding what you wrote’—which in Portuguese is a complete synonym for ‘I have a question regarding what you wrote.’ Not so in English. A near relationship meltdown disaster was narrowly averted when the Brazilian programmer understood the difference and explained it to his counterpart. No more ‘doubts’ were expressed, and the situation progressed smoothly.”

Yet again, the expectations from both sides were completely different, and they were governing the individual’s behaviors, leading them down to unexpected and troublesome results.

Organizational policies that may help manage expectations and therefore cultural differences in distributed software development should support the surfacing of assumptions and expectations by educating stakeholders about the respective culturally and contextually bound differences in expectations about knowledge sharing in the context of software development and the need to make them explicit. When that happens, people’s actions may still not change, but at least the likelihood of unintentionally creating an uncomfortable situation with distant partners decreases.

Such surfacing may be achieved with cultural sensitivity training, and in particular training that is specific enough to be of help in software development tasks. This is exactly where the expectation based approach proposed here shows its strength. For instance, a short workshop can be held (even virtually over a collaboration tool) where team members bare their assumptions and expectations on a very narrow set of task-specific situations. One of the major advantages of such effort is that it is well contained and minimizes the creation of uncomfortable face loss opportunities, since there is no questioning of deep-seated general cultural factors.

As one thinks of culture and context in terms of a number of expectations
related to a particular behavior — in the case of this short “how to” statement, on distributed software development behaviors — it becomes clear that there are many opportunities for misunderstandings across dispersed groups. Unless that problem is tackled up front and in a decisive manner, limitations in performance are likely to be the consequence.
References


CHAPTER 8. MANAGING CULTURAL DIFFERENCES IN GSD...
Part III

Tools and Infrastructure
Chapter 9

Choosing Communication Media for GSD

Tuomas Niinimäki

9.1 Introduction

Why is tool support needed in GSD?

To overcome the challenges caused by distance, GSD projects typically use a variety of communication tools, such as the telephone, teleconferences, electronic mail and instant messaging. These communication tools provide access to different communication media, which each have different properties and abilities to mediate the communication between communicating parties.

This chapter presents the central theories helping to understand, evaluate and choose communication and collaboration tools for distributed software development. The concepts discussed in this chapter can help organizations to understand their requirements and challenges related to communication in distributed teams. As these theories aim at generalizing the findings and phenomena from individuals and teams into commonly applicable patterns and ideas, they may not be able to provide obvious and fast-to-apply improvement ideas for every particular context of software development. However, we believe that these theories provide a strong basis for systematic improvement of communication and collaboration tool support for global software development projects.
CHAPTER 9. CHOOSING COMMUNICATION MEDIA FOR GSD

9.2 Fundamentals of tool support for communication

Theoretic knowledge on choosing, using and evaluating communication and collaboration tools for teams aims at modeling, refining and generalizing the findings and results from research on communicative and collaborative activities within and between teams. Specifically, media choice theories try to explain the reasons why certain tools should be used for a certain task or activity.

In many cases, there are several ways (media) for delivering the same message. These media typically differ in the extent they restrict or constrain the possibilities to express, deliver and present the idea. Such constraints can be either due to the medium’s ability to address different human sensory systems (e.g. audio-only medium vs. text-only medium), to its ability to represent the intended message (e.g. availability of special characters for mathematics or the resolution for graphics), to the message producing capabilities of the medium (e.g. possibility to use moving image), and to the social ”constraints” of a given medium (e.g. availability of a certain medium for senders and recipients).

In the following, we present two central and useful theories for choosing the right tool for different communication situations in software engineering.

9.2.1 Media richness theory

Media richness theory\(^1\) tries to assess what characteristics of a message and its medium lead to successful communication. The theory attributes four characteristics for the communication medium: its capability for immediate feedback, the number of cues and channels available, the language variety, and the degree to which intent is focused on the recipient.

Based on these four properties, media richness theory and related subsequent research suggest that face-to-face communication is the richest medium for communication, followed by video (visual + audio communication) and audio-only communication. Text-only media are considered to be the less rich, or leaner, media for communication.

In addition to a categorization based on a purely sensory basis, the theory also suggest differences in the medium’s richness based on the intended audience for the message: media that are more directed or addressed (e.g. personal letters, or one-to-one discussions) are considered richer than unaddressed media (e.g. bulk letters, reports, or lectures from one-to-many).

\(^1\)Daft and Lengel (1984)
For the message, media richness theory suggests two characteristics to consider: message uncertainty and message equivocality. Message uncertainty is a factor derived from the difference between information already possessed by the individual (or organization) and information required for a certain task or activity. Message equivocality is a concept for message ambiguity, caused by conflicting interpretations about e.g. a situation or environment.

Information richness is the ability of information to change understanding within a time period. Communications that can overcome different frames of reference and clarify ambiguous issues to promote understanding in a timely manner are considered richer, while communications that take a longer time to convey understanding are thought to be less rich. As such, information richness can be considered as a product of the properties of a communication medium and the message delivered through that medium.

Media richness theory suggests that the sender should ideally use the richest possible medium for the message. In reality, however, this is often not possible. In such cases, the sender should consider the purpose and intent of the message, and select the appropriate medium accordingly. On the other hand, it is also understood that a richer medium typically requires much more involvement and attention from the audience. Especially in the context of professional knowledge-intensive organizations, attention is a scarce resource, and this further restricts the availability and feasibility of the richest possible medium for communication. In the selection of medium for a message, the theory specifically suggests that a leaner medium should be used for unequivocal and certain messages, while more abstract, equivocal and uncertain messages should be delivered via richer means.

Media richness theory has its shortcomings, most notably as the view on communication and collaboration taken is task-oriented, taking only the medium and the message into account. In many cases, when working in a distributed cross-functional team, it is necessary to understand the context of communication, the differences in participants’ backgrounds, experience and culture, as well as the individual preferences on communication.
CHAPTER 9. CHOOSING COMMUNICATION MEDIA FOR GSD

Media Richness Theory (Daft and Lengel 1984)

Communication media vary in the richness they are able to provide for communication. Richness is determined by the medium’s capability to immediate feedback, the number of cues and channels available, the language variety, and the degree to which intent is focused on the recipient. Collaboration on equivocal and uncertain tasks should be done via rich medium, ”easier” tasks should be collaborated via lean media.

Match between task and media is supposed to lead to better performance:

<table>
<thead>
<tr>
<th>Too lean medium</th>
<th>Too rich medium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message cannot be delivered effectively as medium cannot express the message</td>
<td>Communication is not efficient due to overcomplication</td>
</tr>
</tbody>
</table>

9.2.2 Media Synchronicity Theory

Media synchronicity theory\(^2\) was built on further research on media richness theory, to overcome the limitations the media richness theory has from the viewpoint of communication and collaboration. Dennis and Valacich define media synchronicity as ”the extent to which a communication environment encourages individuals to work together on the same activity, with the same information, at the same time; i.e. to have a shared focus.”

Media synchronicity theory suggests that effective media use requires a match between media capabilities and the fundamental communication processes needed to perform the task. The theory focuses on two communication processes: conveyance (exchange of information), and convergence (development of a shared meaning for the information).

While many tasks involve both conveying information and converging on shared meanings, a single communication media is often able to either convey information or converge knowledge. The media synchronicity theory suggests that multiple communication media should be used when performing a task.

\(^2\)Dennis and Valacich (1999)
Media Synchronicity Theory (Dennis and Valacich 1999)

Based on two communicative processes: conveyance and convergence of information.

Conveyance = sharing information. Lack of conveyance is potentially manifested as problems:
- People are not aware of important issues and decisions
- Duplicate work
- Delays due to waiting for answers

Convergence = building shared knowledge. Lack of convergence is potentially manifested as problems:
- Miscommunication, misunderstandings
- Doing irrelevant work
- Poor quality of deliverables
- Lack of trust between team members

9.3 Properties of communication and collaboration tools

There are several categorizations for communication and collaboration tool properties, depending on the purpose of the assessment and viewpoint of the evaluator. Media synchronicity theory gives five characteristics for a communication tool, through which one may evaluate its suitability for the two communicative goals defined by the theory, information convergence and conveyance. The five characteristics are:

- Immediacy of feedback
- Symbol variety
- Parallelism
- Rehearsability
- Reprocessability
Table 9.1: Synchronous and asynchronous communication

<table>
<thead>
<tr>
<th>Synchronous</th>
<th>Near-synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both the sender and the recipient must be engaged at and available for communication at the same time.</td>
<td>The tool provides high immediacy of feedback, but also functionality to allow queuing messages for later action.</td>
<td>The sender and the recipient are independent of each other. Communication is possible even when all parties are not available at the same time.</td>
</tr>
</tbody>
</table>

9.3.1 Immediacy of feedback

Immediacy of feedback represents the extent to which the communication tool allows the communicating parties to be aware whether the message was received, understood and acted upon by the recipients. As the tool provides support for feedback from recipient to sender, it helps the sender (the source of information) to verify and evaluate the reception of the message. Through feedback it is possible for the recipient to notify the sender about ambiguous issues in the message, and for the sender to reply with clarifications and to provide additional information. The extent to which a tool provides possibilities to feedback — and the latency between a message and the feedback for the message — define much of the suitability of the communication tool for tasks requiring tight and synchronous collaboration.

Communication media can be divided into two main categories based on the immediacy of feedback they provide: synchronous communication media requires both the sender and the recipient of the communication to be engaged in the communication at the same time, while asynchronous media allow the communicating parties independence to communicate at different times.

However, some of the emerging communication techniques have blurred the distinction between synchronous and asynchronous media, resulting in near-synchronous communication tools having characteristics of both. The best example of such a tool is instant messaging: while it is possible to communicate synchronously over instant messaging, the tool also allows extended delays without excessively disturbing the natural flow of communication.

9.3.2 Symbol variety

The symbol variety of a communication medium determines the possibility to express various messages and meanings over the medium. The concept of symbol variety is closely related to the concept of medium richness, a
property defined in the media richness theory. However, the symbol variety of a communication medium extends the concept of richness in the sense that it focuses also on the variety of symbols within a single medium rather than looking at the variety of additional cues available for conveying message through the communication tool. The effect of a high symbol variety in communicative processes is to make the expression of complex ideas easier, as the medium with the higher symbol variety provides more “building blocks” for constructing the message.

High symbol variety in itself does not always help in expressing the message in the intended way using the medium, if the provided set of symbols — varied as it may be — does not match the symbols required to communicate the message. In software engineering, such a situation would arise when attempting to convey source code lines over verbal communication: while the symbol variety in verbal communication can be considered at least equal to written text (i.e. all written symbols can be expressed verbally), the efficiency is lower, as some specific symbols required by the source code (e.g. curly braces) are more cumbersome to express verbally than in a written form.

The possibility and ease of expressing different messages with the communication tool depends on the symbol variety of the communication medium. Higher symbol variety increases the scope of the communication tool, but can also make comprehending the message more difficult, as high number of symbols increase the number of possible interpretations.

### Symbols and communication

All mediated communication is inherently represented and expressed as symbols. Different media often provide different set of symbols that can be mediated through it, i.e. one medium uses letters, other may use spoken words, and third gestures and facial expressions as symbols. Some messages can be more easily expressed with one set of symbols, while other messages may need another set, or even combining multiple symbol sets.

In all cases, sender and recipient must agree on the interpretation of these symbols in order to succeed in unambiguous communication.
CHAPTER 9. CHOOSING COMMUNICATION MEDIA FOR GSD

9.3.3 Parallelism

Communication media differ in the degree of parallelism they provide for the communicating parties. Some communication media require the full attention of the communicating parties, while another may allow people to perform other activities at the same time, without lowering the efficiency and utility of the communication medium. The degree of parallelism affects the way a communication tool can be integrated into other work activities. For some tasks, e.g. programming, it may be beneficial to be able to communicate with peers and still be able to continue to work on the task, while for other tasks, e.g. planning sessions conducted via a communication medium (e.g. teleconference or video conferencing) it may be more desirable to force every participants’ attention to the task at hand.

Parallelism is also related to the immediacy of feedback of the communication medium. In many cases, a medium providing less immediacy of feedback tends to provide more parallelism. As one expects that reaction from the medium (e.g. answer to a question) typically takes a longer period of time, the medium directs its’ users to work on other tasks in the meantime. As an example, it is rarely sensible to stay idle while waiting a reply to an email message, even when the email message may be of high importance, while even several minutes of idle waiting can be accepted when using telephone as a communication medium.

9.3.4 Rehearsability

Rehearsability is a characteristic of a communication medium that allows the sender time to review, rethink and rephrase the message before sending it to the recipient. The possibility of spending time to review and modify the message privately before sending it helps the sender to fine-tune the expression, and gives extra time to ensure that the message is properly expressed, understandable and consistent.

The ability to rehearse the message makes it easier to produce understandable messages even if language skills would be less than perfect. In addition to allowing time to review spelling and syntax of the message, it also makes it possible to restructure the message, making communication more precise and efficient. As senders may review the communication before sending it, it reduces the barrier to initiate discussion, and makes it easier to engage in on-going exchange of ideas.
9.3.5 Reprocessability

Reprocessability of messages from a communication medium is a characteristic related to rehearsability, but addresses the issue of managing the messages during or after the communication has occurred. Reprocessability represents the extent in which the sender or the recipient is able to revisit the communication event at a later point of time. A communication medium with high reprocessability provides its users with means to review the contents and participants of communication, to process the communicated information, and to relate the message with earlier communications and other sources of information. The ability to do so helps the communicating party to form a better understanding of the nature and meaning of the communication, to acquire and review additional information required to comprehend the intended message, and to reorganize the information into more suitable form for further use. Reprocessability further enhances the ability of a participant in communication to explain the contents of the communication to other parties, who have not taken part in the original communication event.

9.4 Requirements for collaboration tools in GSD

What are the requirements GSD specifically imposes on selecting tools and infrastructure solutions?

Primarily, the distribution of software work introduces the following four main categories ("distances") affecting communication and often hampering it:

- Geographical distance
- Temporal distance
- Cultural distance
- Organizational distance

The effect of these distances on communication and collaboration in GSD projects will be discussed in more detail in following chapters.

9.4.1 Geographical distance

The seminal work by Tom Allen reported that the frequency of communication between engineers would drop radically over a short distance (< 30 m) and that the frequency would remain at the same low level over longer distances, whether the engineers were located 30 meters or miles apart. The physical separation between team members raises the need for mediated
communication between team members, as frequent face-to-face communication is not a feasible option for people working in distinct sites. The decline of face-to-face communication as a viable communication medium puts forth several challenges on planning and implementing communication practices within a distributed team and project.

Firstly, the richness of mediated communication media is less than that of face-to-face communication. The lack of media richness affects different engineering tasks differently: while some tasks can be performed fairly well with leaner communication media, ambiguous tasks, such as planning, solving severe problems or maintaining the sense of togetherness within the team, can be greatly hampered by the lack of additional cues provided by non-mediated communication.

In addition to limitations imposed on communication media richness and synchronicity, physical co-presence provides team members with awareness of many aspects related to their team and project. Overall project awareness — the comprehension about what is going on in the project — can be much more difficult to attain when people are not sharing the same physical space. Physical co-location forces this awareness to team members: whenever there is active discussion within the shared physical workspace, team members are involved in it, whether actively or passively. In distributed projects, there are several practices and tools, which aim for improving the project level awareness, such as task and issue trackers or frequent regular team meetings. Using such instruments adds overhead to the project, but is crucial in implementing project awareness in distributed teams, and thus reduces the risk of duplicate or erroneous work within the project.

In addition to providing informal awareness about the project context and status, physical proximity provides awareness about presence and availability of people. In a shared team room it is easy to evaluate whether a fellow team member is available for interaction. In distributed settings, there is a need for tools providing presence awareness for remote team members. Such tools in a common use usually include instant messaging systems, which allow people to set their own presence status (e.g. available, busy, out of the office) and calendars, which allow others passively check the availability of a team member.

**9.4.2 Temporal distance**

Distributed software development often involves working in different time zones. This temporal distance — distance in time — further limits the possibilities for communication and collaboration between distributed team members. The difference between time zones of different sites decreases the
number of common working hours for both sites, thus constraining the opportunities for having shared, synchronous communications. The limitation on common working hours for all sites affects the daily work by introducing delays, as one party may be requesting information when another party is not working. Moreover, the lower number of common working hours limits the maximum length of common meetings, which in particular affects planning sessions in iterative and incremental software development projects.

"I think the most difficult thing for the development was that we were in two different time zones – completely so. So either people in the States needed to get up early or people in [Europe] needed to stay up late for communication, meetings. [We had to communicate] through mail but also through telephone conferences."

– Developer

As with physical distance, the challenge for providing awareness for the whole team is an important issue, and solving this issue becomes increasingly difficult as temporal difference increases. The less overlap the team members across different sites have in their working hours, the fewer are the opportunities to use synchronous communication within the team. Synchronous communication is especially useful in providing real-time information about the status, progress and context of the project, in addition to being useful in communicating and collaborating across sites instantaneously and on ad-hoc basis.

As the temporal distance limits availability of synchronous communication, the role of asynchronous communication becomes more important. Written agendas and meeting minutes are needed to involve people from distant sites in meetings and decisions, and written specifications and technical documentation are needed at least to some extent. The lack of interactivity as well as the lower richness of asynchronous media will affect both the convergence and conveyance of information within the team, hindering especially the completion of equivocal and uncertain communicative tasks, such as planning and conflict solving. In addition to the purely technical domain, this situation can have considerable effects on teamwork, job satisfaction and personal motivation as well.

9.4.3 Organizational distance

Working with external parties is common in software projects. At least software projects that are customized for an external customer involve — by
CHAPTER 9. CHOOSING COMMUNICATION MEDIA FOR GSD

definition — an external party, the customer. In addition to customers, many projects involve other external parties as well: there can be subcontractors working on behalf of the provider for some specific part of the project or other providers all working for the same customer but needing to coordinate their work to some extent.

Under the term organizational distance, we group all phenomena resulting from working across organizational boundaries, whether they are caused by differing working practices, organizational culture, financial rivalry or other conflicts of interest. While many of these issues will be present when working with an external customer, involving multiple organizations to work with the actual software development is usually even more challenging.

Many distributed software projects may be built through relationships between either two organizationally separate entities, which may belong to the de jure part of the same company structure or be completely separate legal entities. In larger companies, it may not even be relevant whether the two organizations are part of the same company, as the organizational culture may still be different in different branches of the company, and as there might be intra-company rivalry and conflicting interests between the two organizations.

The implications of organizational distance on tool support for distributed software development are manifold. Depending on the level of trust between the organizations — and the measures in place to increase and control the trust — there might be trade secret and intellectual property rights to be respected and protected. In many of the case projects studied, we found a need to hide some information, e.g. resource utilization or technical details, from external organizations. While we can see the benefits of open communication and collaboration between individuals and organizations, we also acknowledge the existence of justified reasons to keep back some information, especially in the early stages of collaboration with a new partner organization.

As organizational boundaries can in many cases give rise to communicative boundaries, there is a need for practices and tools to overcome the negative aspects of these boundaries. Explicitly defined work practices and tools — and also the expectations on how to apply these practices and how to use the tools — are necessary to resolve any existing conflicts and to forestall future problems. It is especially useful to agree on which communication tools to use and how they should be used when communicating over organizational boundaries. Such agreements on tool use may include response-level agreements (e.g. email should be answered within the next working day), and practices related to instant messaging and presence (e.g. "busy" status should be respected).
CHAPTER 9. CHOOSING COMMUNICATION MEDIA FOR GSD

9.4.4 Cultural distance

Cultural distance causes similar challenges to communication in distributed software projects as does organizational distance. Cultural differences manifest themselves as differences in tacit assumptions and expectations, as diverse working practices and varying preferences in communication and collaboration.

The main considerations of cultural distance in selecting and using communication tools in distributed software projects are related to understanding the diversity in work practice preferences, to the varying level of language skills and to the opportunities for supporting convergence with proper selection of communication tools.

The diversity of working practices can be one consequence of working in cross-cultural teams, as different underlying cultures will inevitably affect the way people think and feel about different issues. Some cultural traits may be more present when using certain communication media, while other cultural traits are emphasized in others. As an example, strong social presence provided by a communication tool may increase the effect of power distance as a cultural trait: when working with cultures with differing views on power distance, a tool providing less social presence may help in avoiding conflicts caused by this cultural difference.

"We had many situations where we had a telephone meeting [...] , you have people saying "Yes", and then after the meeting you don’t have the feedback that you expect. [People] had said "yes" either because they had been afraid to ask a question, or afraid to admit they had not understood."

– Manager

The solution to neutralize the negative effects of diversity of communicative behavior in different cultures is to provide the cross-cultural team with multiple media for communication and give them multiple forms of collaboration. In practice, this means acknowledging the need for both text- and audio-based communication, as well as keeping both synchronous and asynchronous communication modes available. Furthermore, awareness of the existence of differences in communication and collaboration preferences within the team will help in starting discussions about potential challenges and in coming up with solutions fitting the actual configuration of the team.

The level of language skills is also an important factor when selecting the communication tools for a distributed software project. In our studies, we have found that if, as a result of a self-assessment, an individual consid-
ers his/her language skills as inadequate, it leads to a preference to use textual communication; this is shown at least in the preferred choice between the use of instant messaging and telephones. We attribute this preference at least partially to the interactivity and rehearsability properties of a communication media: when a person considers his/her language skills to be inadequate, he/she will prefer to use a medium that allows one to rehearse the communication event, i.e. to draft the message first, and then revisit it for possible grammatical or spelling errors, if necessary.

"Because they have a challenge with the language, it’s easier for them to have some extra seconds to write the message and read the clear message from the on-site organization. So they actually enjoy communicating in writing through chats."

– Project manager

On the other hand, we have found that a communication medium supporting interactivity is useful in overcoming the differences in working practices and the underlying assumptions caused by the cultural distance. As the delay between subsequent messages is lower, the possibility for mistakes remaining uncorrected is smaller. The relative ease of communicating via the medium also shortens the messages and increases the number of turns taken in one discussion. This allows an easier way of organizing the concept at hand into smaller pieces, which in turn helps to detect faulty assumptions and misunderstandings within the concept.

9.5 Professional role in media choice

The work role of a person seems to affect the media choice for communication. Our interview data shows that technical personnel — e.g. developers, testers and architects — preferred the use text-based communication media over audio-based communication media.

We found out that even in cases where the technical personnel had the same communication media available as, e.g., the managerial or sales personnel, they usually preferred to use text-based tools when they themselves initiated communication. The preference for text-based tools may be related to several issues specific to their work items and working environment, as well as to their personal qualities.

On the other hand, people in managerial roles often preferred the use of telephones, as hearing other people provides more cues to the communication; some project managers told that they could get much more information
about the project context from the tone and intonation than from the actual words said.

9.6 Communicating technical information

Properties of the task seem to have a strong effect to media choice in the projects studied. Technically oriented tasks were often communicated via a text-based medium. Text-based media in general is well suited for conveying technical, software-related information, as most of this technical information is fully text-based and needs to be distributed in a precise form. The use of lean, text-based medium for such certain and unequivocal information is suggested by the media richness theory, but this finding is also supported by the media synchronicity theory. Furthermore, the main two text-based tools used by the projects, email and instant messaging, were used for slightly different tasks. Instant messaging was preferred for finding out the correct answer for a single technical question, while email was more focused on sharing managerial information and more comprehensive and detailed knowledge about the software. As email messages have more ability to act as stand-alone, independent documents, this medium seems to have the ability to converge information into coherent knowledge.

9.7 Communication modes

In software projects, the need for communication takes many forms. While many development tasks mostly require one-to-one communication, i.e. when finding the correct or preferable answer to a problem, there are several occasions where one-to-many or many-to-many communication is needed. One-to-many communication is commonly needed to share information about the project, while many-to-many communication is often used to come up with preferred solutions and to solve problems. As there are different communication modes for these different communication needs, and available communication tools differ in their support for different modes, the communication mode is an important factor for media choice in software projects.

9.8 Summary

The challenges on communication in globally distributed software development can be roughly divided into four sources, or distances, of distributed software development: geographical, temporal, organizational and cultural.
Various communication and collaboration tools try to overcome the challenges caused by these distances in several ways. Theories on media choice provide useful insights and models on how communication and collaboration tools should be selected and used in distributed work. The properties of communication media determine how well suited the tools are for a particular project context. While the theories can suggest some properties a communication tool should have in a given context, the main benefit of analyzing the tools and situation is the better understanding of both the project context and the possibilities the tools provide for communication. In general, it is considered to be a good practice to use multiple communication media and multiple ways of collaboration in distributed software projects. This ensures both higher efficiency of communication, as well as elimination of communication problems caused by mismatches in working practices and in individual assumptions on communication practices.
References and additional reading


Chapter 10

Communication Tools for GSD projects

Tuomas Niinimäki

10.1 Introduction

This chapter presents the communication and collaboration tools used in the software projects studied. Each section discusses a communication tool, how it was used, and the purposes and the practices related to using the tool. The actual context of communication tool use is illustrated with quotations from the interviews we conducted in the MaPIT research project.

Communication tools can be divided into two categories based on the level of interactivity they allow: synchronous and asynchronous communication tools. While there are several other classifications for them, we believe this division is the most fundamental one when choosing communication tools for a project: it has the largest effect on the way communication occurs, and determines how efficiently the tool can be applied to different team configurations.

10.2 Synchronous communication

Synchronous communication requires all participants of the communication event to be engaged in the communication at the same time. Such communication is useful especially when ambiguous issues are to be dealt with, as such issues often carry both the need for clarification and commonly agreed and defined concepts. Synchronous communication strongly supports convergence processes, enabling team members to build common understanding on complex or otherwise difficult ideas and concepts.
Synchronous communication is also useful in forcing all relevant parties to be aware of important issues; these tools typically require a high level of attention from all participants, and thus the reception and comprehension of a particular message can be assumed by all those involved in the communication.

The synchronous communication media presented here consist of instant messaging, telephone, teleconferencing, video conferencing and desktop sharing.

10.2.1 Instant messaging

In software projects, technical personnel are mostly involved with the actual technical artifacts of a project. These technical artifacts include software source code, configuration files, infrastructure files (e.g. build scripts) and test cases. In many cases, technical artifacts in software projects are both complex and usually difficult to express verbally, as they are typically composed of or referred to by acronyms, keywords and special characters not part of natural spoken languages. In addition, such technical artifacts have high requirements for preciseness, e.g. code lines must be replicated in an exactly correct form in order to retain their validity. Given these constraints, communication via a text-based medium seems a natural choice for technical personnel, as transferring fragments of these artifacts composed of symbols is both highly ineffective and error-prone via audio-only channels.

"For some part of the topics – e.g. technical class names and so on – we have agreed that they should be communicated via instant messaging or email"

– Developer

Many interviewees considered instant messaging a very efficient tool for communication in global software projects. One part of the experienced efficiency of instant messaging seems to be caused by its informality in comparison to email. The messages communicated via instant messaging are short and dependent on each other, making it possible to both read and write them quickly, in comparison to independent, stand-alone email messages. The experienced informality of instant messaging makes it possible to neglect some of the text-based communication etiquette, such as spelling or grammar, further both lowering the overhead associated with the use of that communication tool, and increasing the informal nature of the medium.

In addition, instant messaging delivers messages near simultaneously, further enhancing the perceived synchronicity of the tool, in comparison with
the multiple minute latency of email. Most of the interviewees using instant messaging were satisfied with its ability to deliver messages quickly, and thus they were using it to get quick clarifications on small questions.

"IM is useful in cases you need a quick reply to something, and you know the other one is there."

– Developer

"Chat session is very real-time, so sometimes we are writing quite quickly. It’s good for fast communication, but not for let’s say official requirements. It’s good for a small question"

– Manager

"We use IM a lot. It is easy. It works well for fast and simple questions: what is going on, what are you doing now and why."

– Developer

Instant messaging tools commonly provide means to make one’s presence status available for others. Fundamentally, presence status is a binary option: a user is either online or offline, i.e. either connected to the instant messaging system or not. For users connected to the instant messaging system, the IM tools used in the companies studied provided a quite versatile selection of predefined statuses:

"I don’t know all the statuses there actually are, but at least the most basic, online/offline, not available, do not disturb etc. are there"

– Project manager

Many interviewees were using instant messaging to check availability of their colleagues for communication, either via instant messaging or via other media. The presence awareness allowed them to evaluate whether an immediate response could be anticipated, or whether they should revert to other sources of information in order to complete their tasks. Many interviewees regarded this as a very useful feature, which removed idle waiting thus improving the performance of their communication.
"I use e-mails especially when the other side is not online on IM. If they are online, I use IM."

— Developer

In addition to presence cues provided by the tools, some people in the projects used the status information to negotiate their availability and willingness to communicate. This kind of negotiating took place in the beginning of both instant messaging communication, and communication via other media, such as the phone. The purpose of this kind of negotiating was possibly to make sure that the visible status was in fact correct and that the persons on the other side were actually willing to take part in the conversation.

"If I need to talk to somebody, I always — even if somebody’s sitting right next to me — just ask "Are you busy now". So it’s good to ask the person if maybe she’s doing something very important"

— Product manager

In some projects, the presence status information from the instant messaging tool was used to interpret what other team members were working on. Sometimes this may even have caused delays, as people cannot proceed until they communicate with somebody who then, nevertheless, might be unavailable.

In other cases, switching status to something else than “online” can be deliberately used to introduce a delay or a pause into communication. Instant messaging makes this possible, and thus some people were using their right for ”plausible deniability”:

"In theory, you can see the statuses from IM, for example, if the architect is in a meeting, or not available, or meeting customers. Not everyone is working full time, and they are not always online."

— Developer

"If they [at off-site] don’t know what to do, and I am busy at the moment, unless they have something else to do, they just sit and wait for someone to come online to answer their question”

— Developer
CHAPTER 10. COMMUNICATION TOOLS FOR GSD PROJECTS

"Every time I ask something more difficult, [the team members on another site] switch to ‘away’ status"

– Developer

Instant messaging was also preferred for urgent communication, mostly because its ability to deliver messages quickly and because it provided presence awareness:

"First, if you don’t see the person in IM, and it’s very urgent, you call him. If it’s not urgent, you write an e-mail. But if you can see him, I use it. That’s why it’s called instant messaging"

– Developer

As instant messaging is a lightweight way of communication, it allows communicating while simultaneously working on other tasks. This multi-tasking ability was often used in meetings, both co-located and distributed. In such meetings, it is often very difficult — if not impossible — to use verbal communication (e.g. phone) with people outside the meeting and still attend the meeting itself, without severely disturbing the flow of the meeting.

Instant messaging was used as a way to get answers to questions during the meeting, or to communicate issues discussed during the meeting, without affecting the ongoing meeting too much. However, there were some doubts about whether using IM during the meetings distracted the user’s concentration.

"We use it for simple questions. If you are in the middle of a meeting, [instant messaging] is very unnoticeable way. Or, if you know that someone else is in a meeting, but you need an answer. The recipient may answer to the question, or at least tell when he’s available for discussion"

– Manager

"They can just send a response and send questions, and they can do another job, not only waiting for a response. And when [on-site resources] get the time, they will send a response"

– Developer
CHAPTER 10. COMMUNICATION TOOLS FOR GSD PROJECTS

"I’m afraid — based on the tapping of keyboards — that some people are using IM while in meetings. I hope they are discussing the defects there. It often happens that at the end of the meeting people start to come back to issues reported at the beginning, announcing them as solved”

– Project manager

Instant messaging

Instant messaging is a near-synchronous communication media: it allows both immediate responses and deferring replies to a later time, if needed.

Instant messaging is
- especially useful for asking simple questions.
- considered to be less formal than email
- less intrusive than the telephone

10.2.2 Teleconferencing

Teleconferencing is a common option for communicating in a distributed software project. In the distributed projects we studied, all used teleconferencing at least on a weekly basis. Teleconferencing was a readily available communication medium for all the projects studied, and its usage was straightforward, as all project members were familiar with the technical and social conventions of teleconferencing.

Teleconferencing was mainly used for weekly team meetings, in which all team members reported their progress in, challenges of and new ideas for the project. As such, teleconferences were mainly used for coordinating work within the team, providing team awareness about what other team members and other stakeholders were doing at the moment, resolving technical and social conflicts, as well as collecting new ideas and improvements for the product, project and ways of working in general. In some projects, teleconferencing was used for smaller groups — or sub-teams — to discuss about their activities across sites: the use in those cases was similar as in
CHAPTER 10. COMMUNICATION TOOLS FOR GSD PROJECTS

team-wide meetings, but the formality of meetings was often more relaxed, as the number of people involved was smaller.

In one of the projects studied, the team regularly used a part of the team’s weekly teleconference meeting for non-work related social issues. In practice, each team member was invited to describe their latest experiences on their personal life lately, to share with others what they had done since the previous week or what plans they had for the week ahead. This was found especially useful in building understanding towards the different cultures present in the team, and to provide additional information about the particular working context and environment on the remote site. However, extra care must be taken so that such informal and non-work related part of the meeting will not take too much time, as this kind of informal socialization was considered inefficient by some team members.

The main challenge in teleconferencing was low sound quality. The poor quality made it difficult for teleconference participants to understand what was being said, especially when working with team members whose English skills were not excellent. In many cases, poor sound quality combined with different dialects and language skill levels seemed to render some teleconferences nearly useless. Communication problems due to technical quality of the medium together with cultural differences and differing expectations of behavior may indeed cause communication breakdowns:

"We had a lot of situations where you have a telephone meeting, you are explaining something, you have a lot of nodding or actually people saying "Yes", and then after the meeting you don’t have the feedback that you expect. I mean that persons have said "Yes" either because they have been afraid to ask a question or afraid to admit that they have not understood what was said"

– Developer

Some projects used instant messaging as a side channel to overcome such problems. In case there was a problem in comprehending what was said, instant messaging was used to communicate the main points of the discussion. Instant messaging was also useful for sharing technical information, e.g. exact names of files, classes or configuration items, in teleconference meetings. Using spoken language for communicating such information is often difficult.
Teleconferencing

Teleconferencing was most commonly used for regular team-wide meetings. In some of the project studied, it was also used for meetings with smaller groups of people.

Teleconferencing was found useful for increasing the overall awareness within the team on project status, working context and acute issues, and for building togetherness and trust within the team, by familiarizing team members with the cultural and social backgrounds of other team members. Dissatisfaction in using teleconferencing were related to language challenges, to low audio quality, and to the effects of cultural differences, especially when communicating existing and potential problems in the project.

10.2.3 Telephones

We found out that one-to-one telephone calls were seldom regarded as effective in the sense of communication performance. Many team members considered using telephone very intrusive and interruptive. This was due to both the psychological need to answer any call as well as the lack of multi-task ability of telephone as a medium; telephone conversations often require such large share of constant attention that working on other tasks is not possible at the same time.

"If we chat messages, it’s not wasting their time. If they get on the phone, then both resources, [on-site and off-site], they can’t do any of the work, just speaking on the phone. But if they use chat then it’s more efficient. They can just send a response and send a question and they can do another job, not only waiting for a response.”

– Team leader

Another source of dissatisfaction in telephone communication may have resulted from the topics discussed in telephone calls. The issues discussed over phone calls were often ambiguous, equivocal and uncertain, and managing concepts with such properties often require both time and effort. Many interviewees acknowledged that these issues are hard or impossible to deal with using text-based media, and as such they were satisfied with the avail-
ability of audio-based tool to solve these issues. In urgent or emergency situations, performance required and telephone calls were related to each other: telephone calls were found to be an especially efficient way of communicating urgent issues requiring immediate action. Telephone was also used as a tool to contact people who were often too busy to be reachable through other media.

"Normally through mails, because they are quite busy and in mail you will see the information, and when you have time you read it and answer. And, when it's urgent I'm using telephone”

– Developer

### Telephones

In the projects studied, telephones were not used much for one-to-one communication. The use of voice-over-IP services was more common than using telephones. Non-technical project personnel preferred to use telephones instead of e.g. instant messaging, to gain extra information on the context at the remote site. The cases in which telephone was found especially useful, were urgent situations, which needed quick reaction and timely answers.

### 10.2.4 Video conferencing

In three case projects we studied, there were meeting rooms equipped with video conferencing facilities; these were used regularly in one project, and occasionally in two other projects. The overall quality and usability of video conferencing has increased during the last decade, and in the projects studied the users of video conferencing were fairly satisfied with it.

The main benefit of video conferencing is the richness it adds to the communication. When properly set up, video conferencing allows all participants to see the facial expressions of others, which helps in evaluating whether communication is successful, whether the message has been understood or whether further explanations and discussions are needed. The live video image also allows the participants to see who is in fact talking at the moment, something that can be challenging when working with new people and over lower quality telephone lines.

We have also found that using video conferencing makes meetings better
structured, and the basic meeting practices are more carefully abided by than is the case with audio-only teleconferences. People tend to be more focused on the meeting, and less occupied with other tasks, which may often be problematic in teleconferences. This particular effect of video conferencing arises from the stronger social presence this medium provides, i.e. during the meeting, people have stronger sense of togetherness when communicating by videoconference than when doing so by teleconference.

High utilization levels of the video conferencing meeting rooms formed the main challenge in these projects. It was nearly impossible to use video conferencing rooms for ad-hoc meetings, as the rooms were usually reserved all the time, and reservations had to be made several days beforehand.

"We don’t have that many video conferences now, because if we can book the video conference room for three days from now, we should have solved the issue already by then."

– Project manager

Another common impediment for using video conferencing was still the relatively long setup time for it, and the relatively low reliability of the equipment. In many cases, the preparations for video conferencing took several minutes, and there always was the realistic possibility of failing to setup the videoconference in time. We believe this is related to the stage of the technological development of video conferencing equipment, and at least some of the issues related to the difficulty in setting them up will be solved in future versions of video conferencing equipment.

For instance, in some case projects, there was a possibility to use desktop video conferencing from the employee’s own workstation, and provided that the sites had enough bandwidth, the quality and reliability of desktop video conferencing was considered adequate.

"[Video conferencing is] better than just telephone, because you can see the people, and I think that’s good, and we also use [desktop video conferencing] with a webcam, so that’s also good to have."

– Developer

We believe that the use of video conferencing will increase a lot in the future. In one of the projects studied, video conferencing using web cameras was commonly used to organize daily scrum meetings with all team members. Several teams within the same project held their daily scrum meetings
consequently, in a row, thus lowering the overhead to set up the videoconference individually for each team. The team members were generally satisfied with this practice.

**Video conferencing**

Video conferencing was used for similar purposes as teleconferencing: regular team-wide meetings.

Video conferencing may help to keep the meetings more structured than in teleconferencing, as people tend to be more focused to the meeting when they know that others can see them.

High utilization of specialized video conferencing rooms was a challenge to extend the use of video conferencing in project studied. This will be rectified, as desktop video conferencing becomes more available.

### 10.2.5 Desktop sharing

Desktop sharing is a tool, which provides all participants with the view of a presenter’s desktop. This functionality is often provided in tools for collaboration over network, such as instant messengers and web conferencing software. Desktop sharing is commonly used in meetings, e.g. teleconferences, to provide all participants — not just those co-located with the presenter — with the presentation material, most commonly slides, synchronized with the actual presentation.

Beyond its use in meetings, desktop sharing was used for demonstration purposes in one of the projects studied. The desktop sharing technology enabled team members to demonstrate the functionality of the software to customer remotely, without the need to meet face-to-face or to train on-site personnel on the demonstrated features.

Another use for desktop sharing was training of new team members remotely. Desktop sharing was found useful also in problem solving and in working together with a remote pair.
"But desktop sharing is [...] actively used, I think. Desktop sharing is used during phone calls, when you are actually working on a specific issue, specific case, doing a presentation. It’s convenient to use the desktop sharing."

— Developer

### Desktop sharing

Desktop sharing allows remote participants to see and interact with the desktop of the presenter. Desktop sharing was found useful in many cases, e.g.

- giving presentations in meetings
- demonstrating the software to customer or remote team members
- training new team members
- problem solving, explaining the exact situation in the software

### 10.3 Asynchronous communication

Asynchronous communication media allow the parties to work independently of each other, without the need to be available for communication at the same time. While this somewhat limits the communication by reducing the possibilities to interaction and increasing latencies in the communication, it allows the participants to determine the appropriate time and place for processing the communicated information. Asynchronous communication is especially useful for information conveyance, i.e. sharing knowledge on an issue familiar to all parties, and on a domain consisting of concepts that either can be easily defined and understood or which have been agreed upon earlier by using other communication methods.

In the following chapters, we present two asynchronous communication media, which were commonly used in the projects studied: email and issue trackers. As with the synchronous media, we present the fundamentals of their use, related practices in using these tools, challenges the teams faced when using these tools, and possible solutions for the challenges.
10.3.1 Email

Despite the emergence of other communication media, such as instant messaging, email was the dominant communication media in all the projects studied. Other communication tools had replaced at least some one-to-one communication in these projects; especially developers had adopted instant messaging as a primary communication tool with their peers. Email, however, was effectively used to communicate team-wide issues to all team members, as well as a means of communication between the team and other stakeholders.

The projects studied utilized project-wide mailing lists, which were used to share important issues with all team members at once. Such use of email was considered useful, as long as the number of messages sent was low enough for team members to actually keep track of the sent messages. Another benefit from using mailing lists was that each team member then had a copy of important announcements and decisions, and had the ability to retrieve this information from their mailboxes when needed.

Another practice many of the teams studied used was to send the transcripts of important instant messaging discussions to the whole team. This allowed a part of the team to conduct discussions via instant messaging, and to come up with decisions that might affect the whole team. Whenever such important decision was made or relevant information was disseminated via instant messaging, the log of the discussion would be posted to all relevant team members. The experiences of this practice were mostly positive in the teams using it, as it combined both the flexibility and synchronicity of instant messaging and the asynchronicity and reachability of email.

There are several aspects of email which were thought to make it more efficient than instant messaging. One email message typically contains considerably more information than a single message in an instant messaging session, and thus the volume of emails is typically much lower than the message flow in instant messaging discussion having the same information value. The lower number of distinct messages and higher capacity of a single message reduces the absolute amount of distractions during the communication session, and thus reduces the disturbance, to other work processes, caused by communication. Email messages are more often complete, standalone entities than single messages or even whole conversations in instant messaging.

When compared to messages or transcripts of other communication media, email messages often capture more on the context of communication, and thus become more shareable than messages from other communication media: Email messages can be easily shared with other people, and there is less need to explain the context of the communication. Thus the desired
aspects can be communicated with more care and more comprehensibly. Lastly, many interviewees regarded emails more formal as a medium, and therefore more suitable for permanent storage in a document repository.

We found out that people preferred to use text-based medium for tasks of low equivocality and high certainty, such as status reporting and task assignment. On the other hand, many interviewees reported that they prefer to discuss new ideas or more complex concepts on telephone, to make sure the other people have understood it. These findings are in line with the suggestions from the media richness theory.

"Status reporting works very well [via email], as well as assignment of technical tasks, but if you want something more conceptual, out-of-the-box thinking, it’s better to use telephone”

– Project Manager

A common challenge when using email as a primary communication medium is due to its asynchronicity: it can often be difficult to get answers to your questions in time, especially when all team members are busy working on their own work items. In such cases, many team members resorted into using other media, such as telephones or instant messaging, in order to gain some insights about the context on the remote site, and to build awareness on the presence and availability of their peers.

"One of the main complaints from the [remote] organization has been that if they have a question, they are not answered in due time. I think that if you are working actually with a problem you take a telephone call and you’re not having any feedback from the telephone call, you send a mail and you are not having any response in a day or two or three, so meaning that you are quite lost in the other end. Sometimes you have to make your own assumptions, meaning that you work more inefficiently than if you had been able to communicate with the experienced developers that know the code in the first place.”

– Developer
CHAPTER 10. COMMUNICATION TOOLS FOR GSD PROJECTS

Email

Email was used in all projects studied. While other tools, especially instant messaging, have replaced it in one-to-one communication, it was found especially useful for informing the whole team on non-urgent issues.

Other benefit of using email was in the ability to easily search and revisit important information about the project.

The asynchronicity of email allowed people to communicate even when all parties were not available at the same time.

10.3.2 Issue trackers

Issue trackers provide a way for structured asynchronous communication and collaboration in distributed software projects. The structure provided for collaboration is based on issues — bugs, features or tasks — of the project. Issue trackers allow certain predefined states to be set for these issues, thus creating a workflow model for the management of these issues. In issue tracking systems, the issues are typically assigned to one or multiple team members, who may reassign the issue to other team members after they have performed a certain task related to the issue, or if they believe another team member should be responsible for further progress on that issue. In addition to status management and assignation of issues, issue trackers allow comments and discussion on specific issues. This functionality allows users of issue tracker to easily describe and share relevant information related to a particular issue, and thus provide a view on the history of the specific issue.

Issue trackers can be integrated into other communication media. Typically issue trackers are connected to email, so that email notifications can be sent whenever there are new issues assigned to a team member, or if the status of a followed issue has changed. Some projects have integrated their issue management system into instant messaging, so that team members can instantly be informed on the issues assigned to them. While integration to other communication tools was generally seen as a positive thing, it sometimes caused confusion about what tools should be used for further communication on the issue and how discussions conducted outside the issue management system should be documented:
CHAPTER 10. COMMUNICATION TOOLS FOR GSD PROJECTS

"Usually it goes quite well, when we open issues in the [issue management system]. Sometimes we have complaints from the project manager or other people, that we should use the [issue tracker] for communicating about the issues, as in urgent cases we tend to use instant messaging or email to communicate."

— Developer

"The [issue management system] is quite rigid, so often I just read the email notification I receive from the system, then contact the customer by email or by phone, and just store the last message back into the system"

— Developer

For developers focusing on quality assurance, the issue tracker can easily become the main communication channel with the team. While communicating about issues related to software quality — e.g. announcing new bugs or keeping track of the status of fixes — can be efficiently done via the issue tracker, in many cases multiple communication channels are called for. The use of a single communication medium, especially the use of the relatively lean and asynchronous issue tracker, may disconnect team members mainly relying on these tools from other team members, and introduce undesired isolation within the team, which can manifest itself as unawareness of the overall situation and context of the project.

When reporting a potential problem or a bug in the software, it was found useful — in addition to text-only comments — to include a document or link visualizing the issue. Some individuals were constantly providing step-by-step screenshots to describe how a certain issue could be reproduced. The practice of including screenshots was considered helpful in communicating the issues to other team members, especially to developers who were assigned the responsibility to fix the issue.

"We are trying to be as clear as possible, when opening issues to the issue tracker. We often include documents with screenshots describing the exact steps how to get into the problematic functionality of the software. I should be doing that even more carefully, and in more detail actually."

— Developer

While the issue tracker tools were originally intended for managing and
tracking the maintenance of software and fixing bugs in software, in many projects the scope of their usage has extended beyond this original purpose. Such new uses for these tools include managing on-going tasks of the team, keeping track of proposed new features, and storing design and implementation decisions on particular work items or aspects of the product.

"We use [issue tracker] actually also for [storing the decisions]. Of course they should be logged in the meeting minutes, in action point register we put also that there is a decision. But nowadays we use more and more [issue tracker] to put our action points, so the decisions are also in the [issue tracker]. We can filter the action points for a certain meeting type and use that in the meeting.”

– Project manager

### Issue trackers

Issue trackers provide a structured way of communication and collaboration on specific work items. Issue trackers were used to track the status of defects, tasks and new development ideas. In addition to tracking the status, issue trackers were used to share information and discuss on those specific work items.

### 10.4 Summary

In Figure 10.1, we summarize our conception on the characteristics of the communication tools used in the projects studied, and their respective capabilities to support the two communicative processes defined by the media synchronicity theory: conveyance and convergence. The positioning of individual tools in Figure 10.1 should be considered only relative to other tools in the figure, not as an absolute measure of capabilities of the presented tool.

All projects studied had access to, and had been using communication tools that — based on the media synchronicity theory — support both conveyance and convergence processes. However, in some cases, the projects were using the tools for suboptimal purposes. Especially email was often misused for problem solving tasks requiring synchronous and versatile communication for building common understanding on the often complex issue
Figure 10.1: The use of communication tools in the projects studied to be resolved. Similarly, many projects were using audio conferencing for sharing information on common project related aspects, as well as distributing work — both tasks that could potentially be communicated via other media as well.

There are several communication media available for distributed software teams. In many cases, there are also many tools providing access to these communication media, and these tools vary in features, usability and performance. As one communication medium or tool is often suitable for a limited set of tasks and activities, it is important to have a comprehensive selection of communication tools available. The availability and utilization of multiple communication tools within a team is necessary also to meet the differing preferences and working practices of individual team members, to allow the establishment of efficient and effective information sharing and communication practices\(^1\). Our results indicate, that in cases where the tool usage was suboptimal according to Media Synchronicity Theory, the communicative parties did notice at least some inconvenience: they were either frustrated in the delays and inaccuracies in the replies they received, or in the inefficiency of communication.

When working in a distributed setting, it is important to make sure the communication tools are compatible with the setup, both technically and socially. This includes the harmonization of the tools’ technical aspects, i.e.

\(^1\)See also Chapter 3.10.
CHAPTER 10. COMMUNICATION TOOLS FOR GSD PROJECTS

using compatible versions as well as open, reliable and secure infrastructure, e.g. proper configuration of firewalls, so that tools can be used at all sites or even remotely, if needed. Social compatibility of communication and collaboration tools includes a common agreement and definition on how the tools should be used, e.g. commitment from all team members to answer to emails in due time, or being logged in to instant messaging whenever available for communication. The common agreement on communication practices is also important, e.g. when deciding who should be present in which meetings, where to store important decisions, and whether to inform the whole team about decisions made privately, for example, in instant messaging discussions.

As with all technology, communication tools evolve over time, and new communication media become available. It is especially important in distributed projects to keep track with recent advancements in this area, to constantly evaluate the emerging needs and challenges in communication, to improve existing communication practices, and if necessary, introduce new communication media to the set of selected communication tools.
Chapter 11

Physical Workspaces and Distributed Teams

Peitsa Hynninen

11.1 Introduction

Communication problems are inherent in software projects, and distributed projects are no exception. If anything, communication problems may only be exaggerated by physical, social and temporal distance¹. One of the largest problems with distributed projects is that they do not allow informal discussions — normally held in a cafeteria or at a water-machine — to be conducted. Informal communication usually increases awareness of what is happening in the team. Lack of such communication in distributed teams creates visibility barriers between the sites and keeps the team members at different sites from getting to know each other. This can further raise the barrier for communication and create social issues.

"Communication is the key [...] it’s important [...] it’s the coffee machine steering that you met someone by the coffee machine, and it’s fine for many projects. If there are five people then there is no need to have any meetings and some heavy organisation, but if you are distributed then you should have some rules."

– Off-site project manager

¹Herbsleb and Mockus, 2003
"In my opinion, because there is not enough communication, it follows that there are misunderstandings. It happens to be a different thing to discuss matters face-to-face and it is easier to go to the room next door and ask 'how did this go again?', in there [instant messaging] you don’t bother to write so much about the things, and this way part of the information is not received and then you have misunderstandings."

– On-site tester

There is more discussion on the effects of distribution for communication in Chapter 9. This chapter will concentrate on physical workspaces of distributed teams and on how workspaces affect communication in and between the sites. We will also discuss the kind of workspace solutions that could ease the challenges of distributed communication.

11.2 Challenges

The probability of weekly, informal face-to-face communication between research and development (R&D) engineers has been found to be strongly dependent on their physical separation in the workplace. Longer distances on the same office floor, and to an even greater extent between different floors, dramatically decrease the probability of communication within the first 30 meters. The relationship between distance and communication is presented in Figure 11.1. After the first 30 meters, the probability reaches an asymptotic level which is quite similar regardless whether people are 100 meters or 1000 kilometers apart.  

Because of lack of communication and face-to-face meetings, there is often lack of knowledge about more personal aspects of the team members in the other sites. Social communication and small talk, which comes naturally for collocated teams, is absent in distributed settings.

"Maybe it is because if you are sitting next to somebody, you are talking with him about his personal stuff, but that’s not usual when guys are . . . when there is this distance”

– Off-site project manager

It follows that members of distributed teams often do not personally know those with whom they are working. This may create a number of problems. Doubts may arise about whether the team members in the other

\[2\text{Allen, 1984} \]
site are actually doing the work they are supposed to be doing. Lack of social contact may also make the team members hesitant to contact the other site. Lack of knowledge about skills and competences of others may cause misunderstandings and make others seem rude or ignorant. All this can amount to negative feelings towards team members from the other site and hamper the communication and collaboration unnecessarily. This is why it is important to conduct frequent visits and arrange team building sessions between the different sites. These activities are discussed more thoroughly in Chapter 3.9.

Distribution creates a set of challenges for communication between the sites but also within the site. If the local teams are scattered across the office in the different sites, the team members tend to see each other less often. This creates problems with informal communication similar to the problems between distributed sites. In the solutions part, one solution presented for the problem is collocating teams in different sites to the same room. Thus, the teams at each site would have their own room where to work. However, there exist a number of obstacles for collocating team members working in the team site. Office buildings often have open office space only or small separate rooms for everyone instead of team rooms. If team rooms exist, they may not have enough space for larger teams. The fluctuation of the number of personnel during the life-cycle of a software project may also create problems for using team rooms. Similar problems come up with visitors from other sites. There may not be room for the visitors to work in the
same room with the rest of the team.

Collocating people in the same space also has its disadvantages. There is a risk that co-location leads to increased interruptions and thus to loss of productivity (Mark et al., 2005). Software engineering is, after all, a discipline where one needs to concentrate on implementation tasks and this may be hampered by constant interruptions. According to our results, however, distractions caused by one’s own team members are seen as less harmful, as they may include useful information regarding the project. On the other hand distractions from people who are not members of the project are seen as more distracting and harmful. This is why hot-desking offices, i.e. open-offices where the places are not assigned and people can take any desk which is available, are especially harmful for the communication of the team. Either the team members may not get places near each other, or if they do, they may still be discouraged to communicate, fearing that they might distract people who are not part of their project.

11.3 Solutions

Spontaneous and informal communication plays a crucial role in creating and evaluating new ideas and sharing information quickly between team members. Recent research shows that workspaces where the possibilities for this kind of informal communication are maximized, i.e. work rooms dedicated to a team, can boost the productivity of the team. Working in the same room also makes it possible to decrease the amount mediated communication (e.g. through a team leader) which is also important for the quality of collaboration in the team. Although it is impossible to have the whole team in the same room in GSD settings, it may be convenient to at least have the team members within the same site in the same room. One of the off-site project managers in our cases described how he found that spending time in the same room with his team members made distributing information from on-site easier within the off-site team:

"I’m sitting with developers. It really is — in my mind — quite good because I’m with the guys, I’m able to somehow follow up what are their troubles, and as well when I’m discussing something with on-site, usually they can hear it and they can actually contribute to this discussion quite often."

— Off-site project manager

---

3Teasley et al., 2000
4Hoegl and Gemuenden, 2001
Collocating people in the different sites can reduce communication overheads between the sites. In addition it gives the developers a better opportunity to communicate the impediments experienced by them upwards in the hierarchy as fast as possible. Co-location of team members can also create cohesion among the team members by increasing the amount of face-to-face communication within the team. The solutions presented below are summarized in Table 11.1

**Arrange a team room.** If possible try to arrange a team room for all sites. This will not only help with the communication within the sites, but also will ease the communication between the sites. Remember that besides the team members also project managers and, if the project has them, product owners should sit in the same room. Arrange enough room for possible visitors from the other sites. See also the ”window” to the other site solution.

**Arrange a team space.** If a team room is not an option, you can still try to isolate an area from the office for the team. In case of lack of space you can, for example, group together teams working for the same customer or with the same product. If possible, try to keep a small buffer area between other groups to keep the distractions to a minimum.

**Make an agreement about silent hours.** We observed that one of the teams we studied had, perhaps unconsciously, created a habit of communicating less in the last couple of hours of the working day. In fact, in the last few hours, their communication dropped dramatically. This seemed to help them in the work needing more concentration. If the team is working in the same space, it may be a good idea to make an agreement about the hours when verbal communication is kept to a minimum.

**Arrange temporary team rooms.** In case you have visitors from other sites, but they cannot be seated near the team, arrange a larger space for the duration of the visit. It may be a good idea to reserve an extra desk or two for these kinds of visits already from the beginning of the project. If this is not possible in the normal workspace, you can also reserve a large enough meeting room for the duration of the visit, which is then used as a makeshift team room. The whole team and the visitors should move to the arranged meeting room for the duration of the visit.

”**Window” to remote site.** If you are able to locate the distributed team in the same room, within each of the different sites, you can connect the distributed sites using some kind of video conferencing software. During our project we saw a case where the different sites were constantly in contact using an LCD TV attached to a wall of the team room, and a computer which was running a video conference window, with the other two sites of the project. This way all the sites had a constant visual and audio connection. The project members seemed very satisfied with this kind of setup. The
benefits of this kind of setup include: seeing who is available on the other site, talking "face-to-face”, hearing the informal discussions in the other site, and noticing possible problems. Constant contact can also make it easier for the team members to feel as belonging to a same team.

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Possible solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team members at the same site lack face-to-face communication, because they are scattered across the office (e.g. own room, hot-desking office)</td>
<td>Arrange a team room</td>
</tr>
<tr>
<td>It is not possible to arrange a team room for the team, because the company has only open-office space. Team members cannot freely communicate because they fear that they distract other in a hot-desking office.</td>
<td>Arrange a reserved space for the team from the open-office with a few meters of buffer zone and taller screens for privacy</td>
</tr>
<tr>
<td>Visitors from other sites do not have space near the other team members’ workspace for the duration of their visit</td>
<td>Arrange a meeting room for the whole team for the duration of the visit</td>
</tr>
<tr>
<td>People feel they cannot concentrate in the shared workspace</td>
<td>Make a commitment within the team to have a few hours everyday, at the same time, when verbal communication is kept to a minimum</td>
</tr>
<tr>
<td>There is lack of information about what the team members on the other site are doing or even what they look like</td>
<td>Arrange extra computers, an LCD TV to the wall on both sites, microphones and loudspeaker and set up some video conferencing software, from which you get a &quot;window&quot; to the other site</td>
</tr>
</tbody>
</table>
References and additional reading


Research Methods

A.1 Research project and goals of the study

This workbook is based on research done in the MaPIT research project (Management, Processes, and IT Support for Globally Distributed Software Development) at Aalto University’s School of Science and Technology in the Software Business and Engineering Institute. The project was financed by Tekes (the Finnish Funding Agency for Technology and Innovation) and the participating five companies. The project spanned from 1st January 2007 to 31st March 2010.

The goals of this study were to address the challenges of distributed software development projects and to find solutions and useful practices for managing and working in globally distributed software development. The final goal was to help the Finnish software industry to share this knowledge and employ the proposed practices and processes.

A.2 Multiple-case study

Our research method was qualitative multiple-case study. Case study was chosen as a research method since we wanted to get an in-depth understanding of practices, processes, tools and group relations of globally distributed software development.

All the company participants of the MaPIT project had globally distributed software development and had joined the MaPIT research project since they were interested in improving further their abilities in global software development. Thus, these companies were the natural target group when looking for case projects.

The selection of case projects was done together with the companies. The case selection was done using purposeful sampling, choosing information rich cases. Our researchers gave the company liaison persons instructions for case project selection. We were looking for globally distributed software development projects that would be ongoing at the time of the research. Both
successful and challenged projects were looked for. The liaison persons suggested suitable interesting projects to participate in for this study, and the final selection of the case projects was done by the researchers. Altogether we selected 15 case projects from four MaPIT company participants, as well as one case project from another interested company. Thus, we had a total of 16 case projects from five companies, which are referred to as case companies in this study. The case projects were developing either new software products or customer specific software, or new versions of these. Three of the projects concentrated mainly on maintenance activities and did new development in small scale only. The case companies are presented in Table 1 and case projects in Table 2. Chapters 1–2, 4–7 and 9–11 are based mainly on cases A-L (12 cases) and Chapter 3 on cases M-P (4 cases).

### A.3 Data collection

The main data collection method was semi-structured theme interviews. From each case project we made several interviews, which provided us with an opportunity for data triangulation. The interviews were designed to take approximately two hours, but depending on the interviewee and his or her role, the interview durations varied from 1 hour to 2.5 hours, two hours being the most common length. The majority of the interviewees were software architects, designers, developers and testers, and for each project 1–3 employees with a project or product managerial responsibilities were interviewed. With a few exceptions, when the interviews took place, each of the interviewees worked only in one of the projects analyzed in this study. Five researchers
from the MaPIT project participated in data collection. In each interview there were always two researchers present. The main interviewer led the conversation and asked questions, whereas the other interviewer took notes and asked additional questions. The interviews were relatively loosely structured and conversational in order to maintain adaptability to the roles and individual experiences of the employees in different projects. In the interviews, people described their own experiences of globally distributed collaboration in the project they were involved in, and the general work practices that had been used in that project. The interviews covered twelve themes related to software engineering, e.g. software development process models, communication and collaboration practices between sites, knowledge transfer practices, issues related to trust and identification, and tool usage in the projects. Moreover, specific problems encountered during any of the projects, as well as good practices and ideas for improvement, were discussed. In the interviews, the issues were dealt with both retrospectively and focusing on the current situation during the time of the interviews. Individual experiences during the early phases of distributed collaboration were described by every interviewee. All the interviews were performed during 2007–2009.

A.4 Data analysis

The interviews were transcribed by an outside company. The transcribed interviews were coded by MaPIT researchers into 20 categories using Atlas.ti, a qualitative data analysis software. The created categories partly arose from the data and partly were established in the beginning through discussion with all researchers. The categories included formal and informal communication, roles and responsibilities, process models, knowledge transfer and group relations among others. After this first round of coding, each researcher analyzed the data further from his or her research point of view and, e.g., collected quotations from the data. Based on the analysis, the researchers prepared feedback sessions for all participating companies. During those feedback sessions, the participants were encouraged to ask questions and comment on the results and in this way also to verify that the findings seemed correct to them.
### Table 2: The case projects

<table>
<thead>
<tr>
<th>Domain</th>
<th>On-site</th>
<th>Off-site</th>
<th>Team size</th>
<th>Interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  Enterprise resource management</td>
<td>Finland</td>
<td>India</td>
<td>14 + 25</td>
<td>3 + 1</td>
</tr>
<tr>
<td>B  Information system</td>
<td>Finland</td>
<td>Lithuania</td>
<td>9 + 6</td>
<td>3 + 2</td>
</tr>
<tr>
<td>C  Information system</td>
<td>Finland</td>
<td>Lithuania</td>
<td>60 + 8</td>
<td>4 + 3</td>
</tr>
<tr>
<td>D  Communication system</td>
<td>Finland</td>
<td>Czech Republic</td>
<td>6 + 26</td>
<td>2 + 4</td>
</tr>
<tr>
<td>E  Legal system</td>
<td>Finland</td>
<td>Lithuania</td>
<td>11 + 6</td>
<td>8 + 4</td>
</tr>
<tr>
<td>F  Enterprise resource management</td>
<td>Finland</td>
<td>Czech Republic</td>
<td>17 + 19</td>
<td>5 + 4</td>
</tr>
<tr>
<td>G  Financing</td>
<td>Finland</td>
<td>Czech Republic</td>
<td>20 + 11</td>
<td>5 + 1</td>
</tr>
<tr>
<td>H  Enterprise resource management</td>
<td>Germany</td>
<td>Czech Republic</td>
<td>25 + 43</td>
<td>0 + 4</td>
</tr>
<tr>
<td>I  Enterprise resource management</td>
<td>Norway</td>
<td>Czech Republic, Sweden, Finland</td>
<td>35 + 17</td>
<td>7 + 3 + 1</td>
</tr>
<tr>
<td>J  Engineering software</td>
<td>Finland</td>
<td>Malaysia</td>
<td>45 + 7</td>
<td>5 + 1</td>
</tr>
<tr>
<td>K  Engineering software</td>
<td>Finland</td>
<td>Romania</td>
<td>45 + 5</td>
<td>5 + 5</td>
</tr>
<tr>
<td>L  Engineering software</td>
<td>Finland</td>
<td>Russia</td>
<td>45 + 5</td>
<td>5 + 4</td>
</tr>
<tr>
<td>M  Printing service software</td>
<td>Finland</td>
<td>Latvia, Germany</td>
<td>7 + 3 + 1</td>
<td>4 + 1</td>
</tr>
<tr>
<td>N  Information management tool</td>
<td>Finland</td>
<td>Russia</td>
<td>10 + 6</td>
<td>5 + 2</td>
</tr>
<tr>
<td>O  Enterprise resource management</td>
<td>Norway</td>
<td>Malaysia</td>
<td>20 + 20</td>
<td>4 + 3</td>
</tr>
<tr>
<td>P  Information system</td>
<td>Finland, Norway</td>
<td>Czech Republic</td>
<td>3 + 7 + 4</td>
<td>3 + 2</td>
</tr>
</tbody>
</table>