Distributed Software Development:
Best Practices Examined

Master thesis, 10 credits, INF 101, in Informatics.

Presented: June, 2005
Author: Edvin Lindqvist
Supervisor: Erik Persson
Distributed Software Development: Best Practices Examined

© Edvin Lindqvist

Master thesis presented June, 2005

Size: 40 pages
Supervisor: Erik Persson

Abstract

The area of distributed development is often said to be troublesome with different problems making the work harder. The aim of this study has been to identify work practices that can make distributed development more successful.

A case study approach has been used to identify successful practices. Interviews were used as the primary method for data collection. To get a wide view of the area, the interviews covered different roles in the studied cases.

The most interesting finding is that distributed development should not only be seen as a creator of problems, but also as an enabler of surplus values that are not possible to achieve in co-located development. Several practices have been identified, covering both how to gain the surplus values and how to avoid the problems commonly associated with distributed development.

Key words

Distributed development, global software development, practice, method.
# Table of Contents

1 **INTRODUCTION**

1.1 AIM OF THE STUDY 1
1.2 OBJECTIVES 2
1.3 DELIMITATIONS 2
1.4 THESIS OUTLINE 3

2 **METHOD**

2.1 DATA COLLECTION METHOD 4
2.1.1 SELECTION OF INTERVIEWEES 5
2.1.2 INSTRUMENTATION 5
2.2 ANALYSIS METHOD 5
2.3 REPORTING 6
2.4 ETHICAL CONSIDERATIONS 7
2.4.1 CONSEQUENCES FOR THE INTERVIEWEES 7
2.4.2 CONSEQUENCES FOR THE READER 7
2.4.3 CONSEQUENCES FOR FUTURE RESEARCH 7
2.4.4 FUTURE CONSEQUENCES OF THE RESULTS 7
2.5 SCIENTIFIC QUALITY 8
2.5.1 EFFECT OF PREVIOUS EXPERIENCE 8
2.5.2 RELIABILITY 9

3 **LITERATURE REVIEW**

3.1 CHARACTERISTICS OF A DISTRIBUTED PROJECT 10
3.2 COMMON PROBLEMS IN DISTRIBUTED DEVELOPMENT 11
3.3 REASONS FOR PROBLEMS IN DISTRIBUTED DEVELOPMENT 12
3.4 PROPOSED SOLUTIONS 13
3.4.1 PRACTICES FROM OPEN SOURCE DEVELOPMENT 14
3.5 METHOD-IN-ACTION AND METHOD-IN-CONCEPT 15
3.6 SUMMARY 16
3.6.1 ISSUES TO BE STUDIED IN THE INTERVIEWS 17

4 **RESULTS**

4.1 THE INTERVIEWS 18
4.1.1 CASE DESCRIPTIONS 19
4.2 ATTITUDE TOWARD DISTRIBUTED DEVELOPMENT 20
4.3 MEETING FACE-TO-FACE 20
4.4 KNOWLEDGE MANAGEMENT 22
4.5 COMMUNICATION FLOWS 24
4.6 LEADERSHIP 25
4.7 QUICK PROJECT CHANGES 26
4.8 PRODUCT OWNERSHIP 27
1 Introduction

Software development has not been excluded from the current trend of increasing industrial globalization. Nowadays it is not unusual that development teams in the same project are localized in different cities, or even in different countries. The cause of this trend is a need to reduce costs, increase competition, and use resources from many different geographical locations (Prikładnicki et al., 2003). The trend does not only imply that development is distributed over national borders, but over organizational borders as well. Steadily increasing outsourcing is one of the reasons for this trend (Paasivaara, 2003).

To have the development teams scattered in different places does not always increase efficiency; the contrary is often the case. Herbsleb (2003) indicates that time requirements for distributed development are often more than double in comparison to co-localized development. The main reasons for this are according to Herbsleb difficulties with communication and coordination.

The problems concerning distributed development do not only apply to development for commercial interests. Another area of current interest is Open-Source development, where the developers are often located completely separate from each other and in many cases in different parts of the world (German, 2003).

Software development methods are often used to make development of large and complex systems possible. Can these methods be of any help for distributed development? Kruchten (2004) says that the current methods do not give much help. One example given by Kruchten is CMM (Capability Maturity Model), which is used to measure maturity of the development methods used in a company. Very little is said about communication at CMM level three. Even so, this CMM level is often used as a requirement when global outsourcing is contracted. According to Orlikowski (2002) the organizational practical knowledge is the most important enabler for distributed development, putting even more focus on development processes and methods.

Some research has been done in the area of distributed development during the last years. The available research shows different views of the area. A majority of the available research indicates communication problems as the main issue and makes a more detailed study of different factors, e.g. cultural differences.

1.1 Aim of the study

There are lots of research looking at the effects and problems from development in distributed teams. In spite of this, there are only a few studies on how to accomplish a successful distributed project. The importance of further studies on current methodological approaches was expressed at the ICSE workshop Global Software Development 2003 (Lanubile et al., 2003). Additionally, a research project involving several European universities (Calibre) has recently been initiated. One of the aims of that project is to study the influence of geographic distance between development teams, and how the problems can be managed. The issue is thus of current interest.

There have been two categories of research on distributed development during the last years (Kovács & Paganelli, 2003):
Distributed Software Development: Best Practices Examined
Edvin Lindqvist

- Research on standards and systems (e.g. CASE tools) supporting distributed design and co-operating engineering environments. This category focuses on “what”, i.e. content specifications for products or projects.

- The other research path studies virtual enterprises. The focus is more on social aspects of distributed development, studying how organizational networks can be created and maintained, and on identifying enablers and constraints. Here we find more project management/methodological issues with a greater interest in “how” and “when”.

This study will focus on the latter issues, with an aim to identify project management and methodological best practices, or heuristics, on how to successfully develop software in a geographically distributed project.

1.2 Objectives

Main objective of this study:

*Which specific work practices can make a geographically distributed software development project more successful?*

Intermediate objectives:

- *Establish a definition of geographically distributed development of software.* This creates an important delimitation of the study, clarifying where the result can be generalized.

- *Establish theories concerning differences between co-localized and geographically distributed development.* It is important to know the problems before trying to find a solution to them.

- *Study how the problems are managed in different organizations today.* To establish best practices it is important to identify successful work practices.

- *Examine if current software development methods can manage the special problems concerning distributed development and if not identify the shortcomings.* Solutions that go beyond the methods are probably depending on methodological shortcomings.

1.3 Delimitations

The study will have the following delimitations:

- Only study domestic, commercial software development

- Look at the lifecycle model in its totality, without explicit studies of the integral parts

- Study companies with well established development methods. This will hopefully make it possible to identify methodological shortcomings

- The study has a focus on social aspects of distributed development. Tool support will not be included, unless specifically mentioned in the interviews.
- No theoretical framework will be used to measure the successfulness of the identified practices. The successfulness will be estimated based on the interviewees’ statements.

1.4 Thesis Outline

The thesis has the following structure:

Chapter 1, Introduction, i.e. this chapter, gives an introduction to distributed development and presents the research question.

Chapter 2, Method, gives a detailed description of how the study will be carried out. This includes techniques for data collection and analysis as well as ethical considerations.

Chapter 3, Literature Review, constitutes a literature review that gives a deeper insight into the area of distributed development. The literature review creates a theoretical framework for the study.

Chapter 4, Results, presents analyzed result from the interviews.

Chapter 5, Discussion, presents a discussion on the implications of the results, some thoughts on how the method might have influenced the results, and relates the results to the literature review.

Chapter 6, Conclusion, contains a brief summary of the results and some ideas on future research.
2 Method

This chapter starts with a survey of methods that will be used for data collection and explanations of why these methods can be considered as usable, continuing with a description explaining how the analysis will be carried out. The last part of the chapter contains a discussion on reporting and ethical considerations.

2.1 Data collection method

A qualitative approach will be used in the study. Qualitative methods are ideal for conducting a study of this kind, with a somewhat descriptive direction where the attitude of the interviewees is essential (Kvale, 1997).

An extensive literature review will be used to establish a framework of the area of interest. This kind of literature review is invaluable for getting a background on the area of interest. It is almost unavoidable to do a literature review before establishing a good research question (Yin, 2003). The theoretical framework established during the literature review will later be used as a base when interview questions are created.

Interviews will be used as the main method of data collection. This method should be very suitable in getting a valid answer to the research question. Personal interviews give the possibility to gather intangible information and to get relevant information with a high level of details. The issue of work practices which facilitate distributed development is a kind of explorative research, for which interviews fit very well. Interviews can be seen as teamwork between interviewer and interviewee, where the direction of the interview can be modified depending on previous answers (Kvale, 1997). This highlights the advantage of the interviewer’s previous experience (discussed in section 2.5.1). With a greater understanding of the issues discussed it is possible to steer the interviews in the most interesting directions. The experience of the interviewees will be the central focus of the interviews. The best interviewees would for that reason be persons with a broad experience in the area of distributed software development. Selection of interviewees is further discussed in section 2.1.1.

Interviews where a specific issue is discussed do often give a more useful result if they have some kind of structure (Kvale, 1997). Since I am quite inexperienced in conducting interviews it is of great importance to have a predefined structure as help. An interview guide will be constructed before the interviews to increase the chance of getting the most out of the interviews. It will also create a firmer ground for the analysis.

When interviews are conducted it is important that the integrity of the interviewee is a central issue, without compromising quality of the gathered information (Kvale, 1997). Ethical issues concerning the integrity of the interviewees are discussed in section 2.4.1.

The study will be focused on a few cases. This makes it possible to interview different roles in the same case to get different views of similar issues. Extensive analysis of the different cases will however not be conducted. The case study approach is used primarily to increase structure.
2.1.1 Selection of interviewees

The number of interviews that can be conducted is limited by the fact that only one person will do the interviews. It is reasonable to estimate that six interviews can be conducted. To give the study more width interviews will be conducted in different organizations. The comprehension of the way of working in an organization can be incorrect if only one person is relied on. An attempt will therefore be made to find two suitable individuals to interview in each organization. To get trustworthy results it is important to do a careful selection of which category of roles to interview.

A case study approach will be used by focus on one specific project in each organization. The project should be one that has traits typical for the organization, possibly with some parts being implemented without distributed development. The project should have been running through several steps of the software development lifecycle. This is to render it possible to catch problems from all parts of the lifecycle.

Much effort will be put into getting an as complete picture of the work processes in the organization as possible. Two roles that together can give this are selected; Team leaders or sub-project managers will be used to get a bird’s eye view of the work. Developers will be used to get a more detailed view of the daily work. The interviewed persons should have worked for an extensive time in the studied project and, if possible, have experience of work in both distributed and co-located projects. This should make it possible to get reliable and complete information in the interviews.

To be able to capture an as true picture of the reality as possible it is important to have the confidence of the interviewee (Kvale, 1997). The measures taken to protect their integrity will be presented before each interview as a way to gain confidence.

2.1.2 Instrumentation

A tape recorder together with pen and paper will be used during all interviews. Less focus will be put on taking notes during the interview, since attempts to write down everything will only lead to missed information and distractions. Notes will only be used to document interpretation of such things that are not captured on the tape recordings, e.g. face and hand expressions, and as a help for the interviewer to avoid missing further discussions on interesting subjects.

The recordings of the interviews will be partly transcribed, i.e. irrelevant parts will not be transcribed. This choice is based on the fact that transcribing interviews takes considerable time, which is better used for analysis. The literature review should give enough knowledge for selecting the interesting material.

2.2 Analysis Method

According to Yin (2003) it is important to have a strategy on how to carry out the analysis. An analysis method suitable for this study is pattern matching, where similar patterns are searched for in the collected data, often by visualising the data. One of the aims of this study is to find similarities in ways of working, so pattern matching should be useful. Yin presents two types of pattern matching suitable for this project:
- **Relying on theoretical propositions.** The theoretical propositions indicate what kind of answers we are looking for. It is suitable to use the propositions and objectives as a starting point for the analysis.

- **Rival explanations.** This includes finding a number of different explanations which are suitable for different theories, evaluate the explanations and choose the one that fits the theory best.

In this study the approach for the analysis will be to *rely on the theoretical propositions*. The reason for this is that the study to a certain degree can be seen as explorative. This strategy is suitable for such studies. Working with *rival explanations* would have been a better approach if the study would have a more explicit aim of building theories.

The analytical method Yin (2003) describes as *explanation building* will be used to strengthen the validity of the study. This is a technique for analysis where competing explanations and models are compared, creating a theoretical framework iteratively over time. This is very suitable in a study like this, where heuristics are created.

The use of a system for coding the transcribed interviews is necessary since this will make the information easier to grasp. The interviews will be analysed based on the research question during coding. A transcribed version of the interview will be sent to each interviewee to avoid misunderstandings and get clarifications and details.

Qualitative research can be very demanding in time and resources. It is very important to be well prepared before each interview in order to get as much relevant information as possible. Only information relevant to the research question will be transcribed; the remaining information will be worked with directly from the recordings. This approach will leave more time for analysis of the interviews.

### 2.3 Reporting

It is important to decide early which target group a study is aimed for, and how the readers are supposed to be influenced. This controls the design of the thesis (Miles & Huberman, 1994).

This study is carried out as a master’s thesis. This makes it important to clearly show the readers how the study has been undertaken. This will be done by clearly describing how methods for data collection and analysis have been used as well as describing important choices made during the project.

The thesis is primarily intended for readers from the following groups: other researchers in the area, method developers, and practitioners of distributed software development. The thesis should give these readers a deeper knowledge and understanding of the area of distributed software development. The target groups are interested in a clear and explicit answer to the research question.

It is important to use a language suitable for the reader (Miles & Huberman, 1994). An academic language would probably be suitable for the target groups of this study.

All interviews are conducted and transcribed in Swedish. Since the thesis is written in English, all quotations are translated. Some quotations might be hard to translate in an idiomatic correct way,
something that might mislead the reader. The quotations will mainly be used to strengthen the results, so the risk of this is considered as limited.

2.4 Ethical considerations

To make an estimate on ethical implications can minimize negative ethical consequences (Kvale, 1997). The ethical estimate shall include possible consequences for participants in the study and in future use of the results. The scientific quality of the study is also an important part of this assessment.

2.4.1 Consequences for the Interviewees

The interviews that are conducted will hopefully reveal how work is really performed in the studied organizations. This can possibly give a different picture than what have been decided by the organizational leaders. If it is possible to distinguish a specific individual it is possible that this individual can be threatened by negative consequences because of participation in the study. To avoid this result the interviews will be handled anonymously and citations with differentiating traits will be avoided. Before each interview the measures to protect the anonymity will be presented and everyone will have the chance to agree on the precautions.

2.4.2 Consequences for the Reader

The study is expected to indicate successful work practices in geographically separated teams. It is possible that the reader will use the results of the study to change work practices in his organization. I will for this reason clearly expressed to which extent the result is generalizable to organizations other than the studied.

2.4.3 Consequences for Future Research

This study concerns an area where there is not much available research. It is therefore possible that the results of the study are used as a theoretical base for future research. This makes the requirements on the possibility to judge trustworthiness and generalizability even higher. To make this possible the study must have a good reliability and clearly show to which degree the results are generalizable outside of the studied organizations. A discussion on reliability in this study is presented in section

2.4.4 Future Consequences of the Results

The results from this study can possibly affect the way of working in distributed development projects. It is important to reflect on the possible long-term impact the results can have. If the results indicate that economical advantages can be reached by working in an inhumane way, it can be questioned whether or not the results should be published. Expected results will probably simplify the work in distributed projects, and by doing this decrease stress. The risk of creating an inhuman way of working is thus quite small, but must still be considered.
2.5 Scientific Quality

It is important that scientific research is meaningful and can be justified since there is lots of research in need of resources (Miles & Huberman, 1994). This study can be considered as meaningful, since previous research indicates important problems. There is also an ongoing research project (Calibre) where a number of European universities are cooperating to study among other things distributed development of software. The results from this project can hopefully support the Calibre project.

Kvale (1997) states that scientists shall avoid to identify too closely with the participants of the study, in social science often called “going native”. This could lead to a less critical view of gathered information. It is hard to find a specific method on how to avoid this. The approach on avoiding this kind of influence in this project will be triangulation of information from different sources, especially sources from different organizations. The supervisor will be used as an additional source of control.

The interpretation of the results is to a high degree dependent of context and prior knowledge. Measurements of the quality of the study will therefore be highly subjective. As an attempt to simplify measurements of the quality the procedures and choices will be carefully documented. This will increase the quality and reliability of the study.

2.5.1 Effect of Previous Experience

My previous experience in the area and in some of the studied organizations is important to describe. I will here make an analysis of how my experience can influence the result.

I have previously worked for about four years with different kinds of distributed development of software. My experiences cover programming, requirements specification, development process implementation and team leading, all of which have been done in an environment of geographically distributed projects. My experience in the area gives me an understanding of what kind of problems that are expected. This is for better and for worse, since with greater understanding come expectations and ideas. To avoid influencing the results in a negative way I will put a clear focus on data gathered from the interviews and try to use methods for analysis where influences can be observed.

I have previously worked in two of the organizations that will be studied. This is also both good and bad. I have a greater insight into which persons that are most suitable to interview, as well as which projects the focus should be set on. The negative part is that I probably have some (maybe incorrect) ideas of which the greatest problems in the organizations are. Triangulation with the results from a third organization will be used as a means of overcoming the possible influence.

I have also worked in one of the projects being studied. Even though I would rather have selected a project where I had no prior experience the most suitable project is the one being studied. The effects of this choice of project are however primarily good. One additional interviewee with suitable knowledge will be included (i.e. myself) and I also have detailed knowledge on the way of working and a good understanding of the meaning of different work artefacts. Since I no longer work in the organization I have no great need for the study to give a positive view of the project.
2.5.2 Reliability

Validity and reliability of qualitative studies are mainly dealing with describing that data have been gathered and analysed in a systematic and trustworthy way. Important for all research is the reliability, i.e. the possibility to repeat the study (Miles & Huberman, 1994). Since this is a qualitative study containing interviews, which can be hard to repeat, it is important to describe how the study has been carried out. This will be done in the following way:

- Present my previous experience in the area, and how it can affect the results.
- Clearly describe assumptions and choices. Where they are made, and why.
- Show how questions in the interview guide relates to the research question.
- Describe how data gathered in interviews are analysed and interpreted.
- Describe the choice of interviewees and the possible influence of the choice.

Increased validity might be achieved by using observations and relate these to the interviews. Observations will however not be carried out since the method is quite time consuming and the available time is limited.
3 Literature Review

This chapter reviews literature relevant to the research question. It will be used as a basis for the interview guide.

3.1 Characteristics of a Distributed Project

Before selecting cases to study it is important to define what is meant by distributed development. It is hard to make an exact definition since all projects are different from each other. Instead the characteristics that change when distance is involved will be described as an attempt to identify which projects are suitable to study.

A large part of the available literature on the subject indicates that the main reason for problems in distributed development is communication. Boland & Fitzgerald (2004) clarify this by using the concept of bandwidth as a means of measuring the amount of communication. When the bandwidth is high, as when developers are co-located, the possibilities of communication are great, with lots of informal communication. With a low bandwidth the communication is limited, structured, and formal.

Allen (1977) has made a study on communication patterns of cooperating engineers which indicates that communication suffers greatly from distance. The amounts of communication in relation to distance take the form of a reversed logarithmic curve. If the distance between the offices of two cooperating engineers is greater than 30 meters the communication is just as low as when the engineers are working in different cities. A similar study has been made by Kraut et al. (1988), who look at cooperation between scientists. This study shows that scientists with offices on different floors in the same building are communicating only slightly more than those working in different cities. The importance of communication is indicated by Allen by showing that high performing employees are communicating several times more than low performers, both in man-hours and number of communication instances.

A wider view is given by Evaristo et al. (2004) who mean that the distributedness of a project is depending on several variables. Evaristo et al. present a list of variables which can be used both for indicating distributedness and probable performance of a distributed project. The variables listed are:

- **Type of project.** Starting from scratch vs. maintenance, focus on hardware vs. focus on software etc.

- **Structure.** How well the division of the product into modules fits the structure of the organization.

- **Perceived distance.** Number of possibilities to meet face-to-face.

- **Synchronicity.** Need for synchronization and coordinated behaviour between sites.

- **Complexity.** Complexity of the product being developed.
- **Culture.** Differences in national and organizational culture between development sites.

- **Information systems methodology.** Differences between methods used for development on different sites, e.g. waterfall vs. iterative development.

- **Existence of policies/standards.** Existence of standards has a significant effect on the possibility to maintain project integrity.

- **Level of dispersion.** Perceived distance in cooperating groups.

- **Stakeholders.** More stakeholders increase the distributedness of the project.

It is obvious that a clear definition cannot be created with this large number of variables. The distributedness must instead be individually judged for each case. What is important is that distance and communication are not the only enablers of distributedness, although they do probably have the greatest influence.

### 3.2 Common Problems in Distributed Development

There are several typical reasons for developing software in a distributed environment; new sites becoming part of the company through mergers and acquisitions, regulations requiring local development, allocating development to where the market is, the possibility of round-the-clock development, and creating a more flexible development organization (Herbsleb, 2000 and Orlikowski, 2002).

The reasons stated above for working in a distributed way can often create competitive advantage for a company, but it is important to remember that there are often problems involved in working in a distributed environment. The most important and common problems are listed below together with a short selection of reasons.

- **More bugs.** A tendency among developers not to confirm assumptions with remote sites creates misunderstandings, leading to errors in the developed software (Herbsleb et al., 2000).

- **Increased cost.** The increased amount of bugs is not the only reason costs increase. The development costs are also affected by a greater need for travelling to remote sites (Orlikowski, 2002).

- **Increased calendar time.** Additional waits are included since it often takes longer time to get answers from remote sites (Herbsleb et al., 2000).

- **Decreased motivation.** Less informal communication makes it hard to create trust for remote colleagues and less feedback is lowering motivation (Boland & Fitzgerald, 2004).

A very clear example of how these problems emerge in distributed development is given by Boland & Fitzgerald (2004). They have studied the effects when the members of an earlier co-located organization were divided in two groups located on different continents. Since they had worked together for a long time it was expected that there would not be much problems. This was not the case, and even though they had lots of earlier experiences together all of the problems above could be seen, with a decreased productivity as the consequence.
3.3 Reasons for Problems in Distributed Development

A more thorough elaboration on the reasons behind problems with distributed development is presented below.

The design and development of large and complex systems is a social process, which can be hard to manage even in a co-located environment. The greatest challenges with software development in general are communication and coordination. This can for example be seen in the need to create a common view of the problem, of the domain, and of the design, the division of work in delimited subtasks, managing changes, and keeping the design consistent in all parts of the system. The difficulties with communication and coordination in a distributed environment make all of these even harder to achieve. The ability of a team or an organization to communicate and coordinate has a great effect on the ability to manage challenges in a distributed environment (Herbsleb et al., 1995).

Since coordination is depending a lot on communication a review of the reasons for communication problems will first be made.

Informal communication is hard to maintain between remote sites (Herbsleb & Grinter, 1999, Boland & Fitzgerald, 2004, Pyysiäinen, 2003). Informal communication is much depending on unplanned contact like “water-cooler-talk” and somebody just “dropping by” the office. The problem is that a lot of communication around the development usually takes place in these forums (Herbsleb, 1999). A large part of the informal communication is cognitive synchronization, i.e. checking that your knowledge or view of an issue is correct. The lack of cognitive synchronization often leads to errors and misinterpretations, resulting in bugs. The importance of cognitive synchronization and similar types of ad hoc collaboration is clearly indicated in the preliminary results from a study by Cherry & Robillard (2004) where developers spent about half of their time on this kind of informal communication. Informal communication is also important for feedback. Boland & Fitzgerald (2004) show that the degree of feedback is lower for distributed development than for co-located, leading to decreased motivation among developers.

More formal types of communication are also affected by some problems from geographical distance. The mental cost of initiating contact is higher for distributed colleagues. There are several reasons. First, it is sometimes hard to know who to talk with. Second, it is hard to know if the person is available. Third, different time zones or different schedules for breaks make it hard to reach the person when it is needed. Fourth, different organizational or national cultures can make it harder to understand each other. The cost of initiating contact results in that you rather take a chance than try to contact someone at a remote site (Herbsleb & Grinter, 1999). The richness that can be achieved in face-to-face communication is also lost when communicating between remote sites, which can easily lead to misunderstandings (Boland & Fitzgerald, 2004).

Much of the communication with remote sites tends to be asynchronous, e.g. by mail. This is probably because the cost of initiating contact by phone is greater than by mail. Using asynchronous communication often creates delays (Herbsleb et al., 2000). One of the reasons for this is that you often prioritize questions from a co-located colleague prior to one from a relatively unknown remote colleague (Herbsleb & Grinter, 1999).

Trust is an important part of communication. If trust exists it makes the communication flow easier and the information is easier to believe, it is also easier to aim at the same goal, and misunderstandings do not lead to as much problems. This indicated that trust can be an important enabler for distributed development. The problem is that trust requires informal communication to be built and maintained (Pyysiäinen, 2003). An unequal division of power between the parties
involved makes it even harder to build trust since this can lead to a feeling of powerlessness and frustration (Kiel, 2003).

A large part of the coordination in software development projects takes place using informal means of communication. One reason for this is that formal communication is best suited for recurrent and well-structured coordination. Informal communication is more suitable when there are uncertainties, which is quite common in software development projects. A high degree of coordination is needed when the level of interdependence between the modules is high, which makes it important with an as low interdependence as possible in distributed projects, especially since inter-organizational projects tend to be less coordinated than intra-organizational (Kraut & Streeter, 1995).

3.4 Proposed solutions

An overview of different proposed solutions on how to be more successful in working with distributed development is presented here.

Herbsleb & Grinter (1999) recommend that the organization is changed to reflect the structure of the system being developed. This helps minimizing need for coordination and communication between development sites. Most questions being raised concern the inside of a module, which can often be answered locally. The interfaces between modules are at the same time quite stable, and the needed coordination can be managed by a few individuals. Grinter et al. (1999) confirm that the way of organizing have a large impact on the result and study advantages and shortcomings of different ways of organizing. Four different ways of organizing are studied; functional area of expertise, based on product structure (as Herbsleb & Grinter proposes), functional process steps, and a special type of product structure with a stable core and subcontracting of modules (customization). The result shows that no way of organizing is best in all cases; each type has its advantages.¹

Orikowski (2002) means that values and knowledge in an organization has a great impact on successful development. Instead of indicating product structure as the most important enabler she says that organizational values have a great part in the success of distributed development. If the values are supporting distribution it will be likely to succeed. Orlikowski has made a study in an organization where the values are seen as a large part of the success of the whole organization, including the highly distributed work. The values identified as important are the following:

- **Sharing identity: knowing the organization.** Creating individual identification with the organization and using a common orientation in the work (“The Kappa way”), globally.

¹ **Functional area of expertise.** Each area of expertise is located at a single site. Efficient in building expert knowledge but requires massive coordination in operational work.

**Product structure.** The organization structure is based on the product structure. Gives the sites a large degree of freedom in how to work but requires clear requirements and often gives a painful integration.

**Process steps.** Organized by speciality with focus on the handover between process steps. Efficient in sharing critical resources but timely handovers and an equal view of the development process are critical.

**Customization.** An architectural core is kept in the main organization and the subparts are distributed. Lets the development of the parts be made close to the customer but requires timely handovers, good documentation and a suitable division of the product.
- Interacting face-to-face: knowing the players in the game. Supporting the building of personal networks and creating a common value that sharing information is important.

- Aligning effort: knowing how to coordinate across time and space. Using common models and methods to support cooperation and coordination, using internal contracts to clarify assignments.

- Learning by doing: knowing how to develop capabilities. Investing in individual development through ongoing training, mentoring employees to advance in their careers, rewarding effort and not criticizing or punishing errors.

- Supporting participation: knowing how to innovate. Creating innovation from broad participation by globally distributed development, involving participants in project decisions, supporting assignments abroad.

Another approach with similarities to some of the values of Orlikowski (2002) is proposed by Pyysiäinen (2003), who recommends trust building as a means of enabling better communication. The best ways to build trust is by meeting face-to-face. This should be done by kick-off meetings when new subprojects are started, and by working together on challenging problems. This is confirmed by Herbsleb & Grinter (1999), who mean that the budget for travels needs to be high. Pyysiäinen says that building trust can also be facilitated by communicating which roles the different persons on remote sites have in order to minimize confusion and create a feeling of control.

Paasivaara & Lassenius (2004) and Bass & Paulish (2004) recommend the usage of an iterative or incremental approach to facilitate distributed development. Revolutional (big-bang) development has a large degree of communication only during integration, when lots of errors usually occur. By using short iterations, one needs to communicate more often, but on relatively limited issues. Short iterations also help in creating a higher degree of coordination in the work, and the quick feedback help in eliminating misunderstandings. While short iterations facilitate communication, they require lots of communication. Since communication often is the greatest problem in distributed development it is possible that it sometimes can be troublesome to use short iterations. Even so, it can be a good alternative when it works. Herbsleb et al. (1995) are further clarifying this by showing that different methods give different levels of communication in different phases of the development. This indicates that it is probably best to select development method based on the prerequisites for each case.

Kraut & Streeter (1995) are pointing out that while it is necessary for developers to communicate more with external partners they also have limited time and are often already at a high level of stress. It is therefore important to find a way to communicate more efficiently, both with local and external partners.

### 3.4.1 Practices from Open Source Development

Open Source software is often said to be a successful and important competitor to proprietary software. The main reasons stated for the success are quick development and high quality. A common belief is that the practices behind Open Source might be successfully applied to proprietary development as well.

The fact is that many of the successful practices from Open Source, e.g. frequent builds, have already been implemented in proprietary development (Fuggetta, 2003). Paulson et al. (2004)
have shown in a study that proprietary development often have the same speed as its Open Source counterpart, so the belief of increased speed might be a myth.

Much of the success for the Open Source community is from its ability to attract developers. The developers are mainly attracted by the fact that they can work in any project they want and can create software that suits their individual needs. This freedom of choice creates motivation\(^2\), but at the same time makes it harder to steer the development towards detailed goals. There often exist some common higher level goals, but the developers themselves determine the detailed requirements. The lack of control makes it hard to predict when a specific function will be implemented, since it depends much on the motivation of a specific developer (Augustin et al., 2002, Persson et al., 2005; Crowston, 2002).

Face-to-face communication is almost negligible in Open Source projects. The communication is instead taking place primarily on the Internet. Formal issues are managed in some kind of issue or change management system. The informal communication takes place in different mailing lists. As for proprietary development, much of the coordination takes place using informal communication (Persson et al., 2005). The hardness of direct communication means that a common vision of what to develop is very important. The most successful Open Source projects have been developing software for which requirements are commonly understood, such as operating systems and web servers (Fuggetta 2003). Areas where only a very small base of developers has good insight, like business specific areas, is thus not as suitable for Open Source development (Crowston, 2002).

The problems with requirements and with creating novel products make it hard for commercial organizations to use the Open Source style of development. Even though many of the practices have already been incorporated into proprietary development there are some things that can be learned from Open Source. There exist some values in the Open Source community that could be useful also to proprietary development; notably a culture of sharing code and knowledge as opposed to the not invented here attitude often seen in commercial organizations and the idea that peerage creates values for all parties as compared to commercial organizations which often foster competition and individuality (Augustin et al., 2002). Open Source also shows how important motivation is to developers, as a motivated developer is willing to work even for free. This is a fact that should be noticed by commercial development organizations that have motivational problems among their employees (Fuggetta, 2003).

### 3.5 Method-in-Action and Method-in-Concept

It is commonly agreed that some sort of software development method is needed to enable development of software in an economically feasible way. Many of the existing methods are based

\(^2\) Achtenhagen (2002) identifies four different groups of Open Source developers, driven by different motivations:

- **IT professionals** driven by a belief that Open Source can provide the best solution for their company.
- **Non-organizational IT experts** wanting to increase their skills in software development and create a reputation in the community and for future employments. Fun is also an important source of motivation for this group.
- **Fun** drives many teenagers to contribute, as well as the possibility to get the software they need for free.
- **Ideologically motivated developers** with the belief that source code should be open. Some of these also have an urge to “bash Microsoft”.

---

15
on successful best practices from several development projects, without any real theoretical background. There are also methods developed on only a theoretical base. In some cases, but sadly not all, the method is tested and evaluated in real projects before release (Fitzgerald et al., 2002).

Some developers use the methods in their completeness and without bypassing any step. This creates lots of extra work since the methods are created to be general and work in different contexts. Some activities might for example only be necessary for large projects. Because of this methods are often specially tailored for each project. The tailoring creates a new method with the aim of doing exactly what is needed in the project (Fitzgerald et al., 2002). The method is going through one final change when used in action. The developers interpret how the tailored method should be used and use it as they believe is the best way. This is called method-in-action (Lings & Lundell, 2004).

Lings & Lundell (2004) describes The Stakeholder Triangle (Figure 3.1) which illustrates the tension between the method developer’s intentions in the method, how the method is really used, and the support of the method in CASE-tools. What is important for this study is the idea behind the possible tension between method-in-action and method-in-concept. By trying to identify the tension it is possible to get a view of the deviations that have been made from the method. This can be used as a means to identify shortcomings of methods.

A view similar to the difference between method-in-action and method-in-concept is presented by Orlikowski (2002) who compares different kinds of knowledge. Theoretical knowledge is similar to method-in-concept and practical knowledge (know-how) can be compared to method-in-action. The ability to work distributedly is according to Orlikowski primarily depending on organizational know-how, not on tool support and similar factors. To be successful in distributed development the organizational knowledge and values must support this way of developing.

3.6 Summary

By using the definition of distributedness it should be possible to select projects that are truly distributed for the interviews. It also creates the possibility of studying to which degree each project is distributed. This can be important for judging the generalizability of the study.

Two different types of solutions can be seen among the proposed solutions.

- Minimizing the need of communication
- Simplify communication

A combination of the two would probably be the best solution. Partly minimizing unnecessary communication, and partly enabling a richer communication.

It can be an important source of success to keep the developers highly motivated, both for distributed and co-located projects. Since there often are problems with motivation in distributed
development, it would be interesting to identify how to facilitate motivation in distributed projects.

Influences from Open Source can also be valuable. Distributed development is often affected by bad motivation among developers. Since Open Source development is highly distributed and the developers often very motivated it can probably be useful to use the underlying sources of motivation from Open Source. Orlikowski (2002) has also seen that an organization that has a culture with similar values as the Open Source community can be very successful in distributed development.

3.6.1 Issues to be studied in the interviews

Each interview will comprise a specific project. A number of issues related to the projects will be examined in the interview. It is important to note that the interviewees are encouraged to include also experiences from other projects.

The following main areas will be examined in the interviews:

The Project. It is important to assess the distributedness of each of the projects, or cases, which are studied. This will help in determining how distance affects the different issues studied for each project.

Product structure vs. organizational structure. It is interesting to study which impact the division of the product over the organization has had.

Power. An unequal division of power among the sites involved in the project could lead to problems with distrust. This might affect the distributed work in the studied projects.

Communication. The largest problem in distributed development is often communication. An attempt will be made to identify which communication problems the interviewees have experienced and their experiences in how these issues can be managed.

Ways of working. An aim of this study is to identify how different methodological work practices affect the work in a distributed environment. Specific practices might give great advantages or disadvantages in distributed development. An attempt will also be made to identify how the organizations have altered the methods to better support distributed work.

Organizational culture. Some of the values in the organization might affect the possibility to be successful in distributed work. An attempt will be made to identify such values.

These issues are further detailed and phrased as interview questions in the interview guide (Appendix A).
4 Results

This chapter will present the results from the interviews. The interviews have first been transcribed, coded and analyzed as described in chapter 2. The interviews tried to capture the experience of the interviewed developers, with the cases used primarily as a way to get several perspectives from the same area. This makes it hard to present a survey of the data gathered in the interviews without presenting the transcribed interviews in their entirety. An unanalyzed overview of the empirically gathered material is for that reason not presented.

It turned out that the analysis of the interviews could not be based only on the issues identified as interesting in section 3.6.1. Some of these were not as important as initially believed and some other issues of great interest were revealed. Seven different recurring themes were identified as most interesting. The themes mainly concern different problem areas in distributed development. The analysis was based on these themes, and the result will be presented using the themes as structure:

- Meeting face-to-face
- Management of knowledge
- Communication flows
- Leadership
- Quick project changes
- Product ownership
- Methodological issues

An insight into the interviewees’ attitude toward distributed development will also be given.

4.1 The Interviews

Six interviews were planned, but only five were conducted. One was cancelled in the last minute and could not be rescheduled in time.

The interviews were conducted at the workplace of each interviewee. Each interview lasted for about 1-1.5 hour.

Roles of the persons interviewed are displayed in Table 1.
4.1.1 Case Descriptions

The interviews were focused on two cases. A description of relevant information from the cases and the organizations involved is presented here:

**Ericsson Microwave Systems**

Ericsson Microwave Systems is divided over several sites in both Sweden and internationally. The largest site is located in Gothenburg.

The case under study is a fairly large project with totally more than 100 persons involved, located at two sites. The project was in the delivery phase when the study was conducted. Interviews were conducted on two sites: Gothenburg and Skövde. All interviewees had worked together with distributed development prior to this project.

**Saab Combitech Systems**

Saab Combitech Systems has offices in several sites in Sweden.

About 50 persons were involved in the studied project. The project was pursued at about six sites during its most intense period. The project was in its maintenance phase when the study was conducted, with only a few developers still involved. The interviews were conducted at the main site of the project: Jönköping. None of the interviewees had any previous experience of distributed development before this project.

**Distributedness of the cases**

Both cases have quite similar traits concerning distributedness. Both have a component structure, where components have been assigned to different sites. The distance between sites is quite similar in both cases. Both are dealing with development in a domestic and intra-organizational context with a great majority of native Swedish speakers, so the culture should be quite similar at all sites. Iterative development methods are used in both cases. The greatest difference that can be seen from the interview data is that the Saab Combitech case had more sites involved in the development.

Based on the similarities it is reasonable to estimate that the level of distributedness is similar for both the cases. This implies that the cases can be threatened in a similar way in the analysis.
4.2 Attitude toward Distributed Development

Two different attitudes toward distributed development could be seen in the interviews. One side believed that distributed development was necessary but should be avoided if possible. The other side thought of distributed development as a means to give the organization surplus values. An example of the latter is given below:

*It is possible to achieve positive effects if you use it in the right way. You can use it to gain a go in a development project. By creating interdependencies, it can grow to something bigger. … Most common is that people don’t have enough time or aren’t good enough leaders, which will cause problems, because it requires much more effort to get such a co-operation to work.*

The possibility to achieve positive effects that are not possible in co-located development is an important observation. Distributed development is often described as necessary but as filled with problems and most literature focus on how to avoid the problems involved. This gives rise to two important issues:

- How to avoid the problems often seen in distributed development
- How to achieve and use emergent effects from distributed development

Since both issues are important they will both be further elaborated in the presentation of the different themes.

An opinion presented by one of the interviewees is that it is not to be taken for granted that it is so much easier to work co-locatedly, especially not if a prolonged relation has been built between the sites. The same is true for distance. It can be as hard cooperating from different offices in the same city as if they were separated by more than 100 km.

4.3 Meeting Face-to-Face

Meeting the other developers is very important for success in distributed development. It is both easier and more fun to work with someone you know. The cooperation with someone you do not know is hindered by not knowing how the other person reacts and communicates. The best way to learn to know someone is by meeting face-to-face. This makes it really important to meet a lot in the beginning of a project. To maintain the contact it is important to meet regularly, although not necessarily as often as initially.

Several interesting issues concerning meeting and travelling were discussed:

*Avoid working alone*

Working alone at one site is not a good solution. It is necessary to have someone to discuss problems and share expert knowledge with. It can also be boring to work alone and stressing to know that nothing will be done if you should be ill.

---

3 All quotations are translated from Swedish.
Optimize travelling

There are some disadvantages in travelling to other sites: it is costly in both time and money and it is often hard to work efficiently when all meetings are over. This makes it important to use the time at other sites as efficiently as possible. An example of how this can be done was given: schedule group meetings with the entire project when several persons from other sites are visiting. These meetings can both be a way of getting to know the other persons in the project and to get a view of the project in its entirety. To avoid travelling only to attend such meetings, a conference telephone can be used to include also those not attending the meeting in person, although the result from this is not as rewarding as being present at the meeting.

Travelling affected by culture

There are seldom problems with gaining permission to travel from the leaders in the organization when it is necessary. Instead it is the developers that are minimizing travelling. Issues requiring travelling are often considered as less important, despite the fact that a possible resulting delay in the project is far more expensive than the cost of the travel. The problem is illustrated below:

To take a short meeting, or take a seat in someone's room when you have a problem is something you do all the time. But the problem needs to be at least ten times worse to order a train ticket. ... A train ticket is nothing compared to a delay in the project. It's not even in the same galaxy of costs.

The problem is often also a lack of willingness to travel. It is often much more convenient to stay at home.

It is often the developers from the small sites that travel to the larger sites, but that is not enough. Everybody needs to travel every now and then to get a better feeling of what is going on at the collaborating sites. The travelling must also include managers and project leaders. How could they otherwise take their responsibility?

I'm convinced that the managers preceding me hardly knew what was going on, or which resources were working with their products on the remote site. I believe that is nonchalant.

Cultural differences

An advantage of working in a distributed environment is that you learn to know people from different parts of the organization, something that is important to bind the company together. But it can also cause problems. An organization that is not used to work in a distributed manner probably has a somewhat different culture and different rules on the different sites. When a distributed project is initiated and the developers start meeting they will most likely see the differences and get the additional workload of trying to figure out what rules that are the valid ones. To gain efficiency and reduce problems it is important that the project leader defines which rules are valid for the project, regardless of which site you belongs to.

Maintaining informal communication

Informal communication easily suffers in distributed development. For this reason it is important to meet often to create informal channels for communication, especially if uncertainty is large in the project. You often get to know a bit more when you meet someone in person than on phone.

---

4 The term leader refers to team leader, project leader, or manager. The term is used in this comprehending way since different organizations give leaders different responsibility.
If the communication can be kept rich with clear paths for communication there are great possibilities of efficient work. To attain this, it is important that everyone takes the responsibility of maintaining communication. The one with the main responsibility and the possibility to create the needed channels is the leader, who must influence the co-workers to initiate contact and create lasting communication channels. A possible outcome of failed communication is illustrated below:

*If you don’t meet each other for half a year and then starts delivering, there will be lots of interface problems all the time. It won’t take long before the developers start cursing the other site: “Why can’t they follow our rules?!?” I would say it occurs at both sites. The motivation will decrease and the project will enter a state of crisis. Work stops and it will cost terrible amounts of money.*

One important thing to keep in mind is that the frequency of face-to-face meetings will change during the project. The contact must be kept also when meetings are less frequent to avoid that the relation falls to pieces.

**Suitable amount of travels**

Meeting each other often is good, but the travels can easily become strenuous for those traveling a lot. The amount of traveling that can be managed is probably depending on distance and personal conditions. Several of the interviewees mention two days a week as a reasonable amount of travels in phases of the project when needs for communication are great.

Table 2 gives a summary of issues concerning meeting.

<table>
<thead>
<tr>
<th>Summary of Meeting Face-to-Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is very important to meet the project members at other sites.</td>
</tr>
<tr>
<td>There should never be only one person working in the project at a site.</td>
</tr>
<tr>
<td>Plan the travels to be as efficient as possible, they are costly.</td>
</tr>
<tr>
<td>Motivate the developers not to depreciate the issues demanding travels.</td>
</tr>
<tr>
<td>It is important that also leaders travel to the remote sites.</td>
</tr>
<tr>
<td>Make sure there are similar rules for all members of the project, at all sites.</td>
</tr>
<tr>
<td>Create channels for informal communication and maintain them also when communication needs are small.</td>
</tr>
<tr>
<td>Be aware that too much traveling is a strain on the developers.</td>
</tr>
</tbody>
</table>

Table 2. Summary of practices and problems from Meeting Face-to-Face

### 4.4 Knowledge Management

Finding the right knowledge can be a challenge in co-located development, but for distributed development it is even harder. The knowledge can be either partly or completely located at a remote site. In your own office you often have an idea of whom to ask and can get an explanation instantly. On a remote site on the other hand, the knowledge might be in the head of someone you does not even know exists. The problems in both cases are illustrated below:
It is always easier if the knowledge exists in the room next door, instead of 150 km away. You also have to use the knowledge next door, something we aren’t successful with all the time since we might not even know that it exists next door.

A way of increasing the awareness of knowledge is to take a couple of minutes and tell the people you speak with what is going on right now. This is a simple and inexpensive way of increasing the awareness. Even so, this is very easy to depreciate since it is easier to see only the time you lose right now, not what you will gain in the future.

**Project knowledge**

It is important to impart a general picture of the project to all participants early in the project, and after that keep everyone updated with major developments. This includes such pieces of information as where different types of information can be found and which rules are used in the project. This is also an important means to be able to quickly integrate new project members. Project rules must be continuously developed and must not differ between the sites. A way of achieving this is to have a special team within the project with responsibility of maintaining the rules.

An up-to-date knowledge database is an efficient way of building and disseminating knowledge, e.g. modelling rules and suggestions on how to use different tools in the most efficient way. It is important to remember that it is not possible to assimilate all kinds of knowledge by just reading a document; sometimes a dialog is required. It is hence important that the individuals with the knowledge take the time to explain to others.

**Avoid hiding information**

Small sites are often eager to create their own profile. This is a way to gain a *raison d’être*. To achieve a competitive advantage the site can keep unique knowledge to itself. The problem is that nobody will even know that the knowledge exists and it will consequently be used in a smaller degree, or even not at all.

A good way to disseminate the expert knowledge from the sites is by arranging recurrent meetings with managers from the different sites and inform them on what is going on in the projects, in cooperation with universities etc.

Table 3 summarizes issues concerning management of knowledge.

<table>
<thead>
<tr>
<th>Summary of Knowledge Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Give a brief presentation of what is currently happening to those you meet to increase the awareness of knowledge.</td>
</tr>
<tr>
<td>Make general information of the project and important issues available to all developers.</td>
</tr>
<tr>
<td>Do not hide information to gain advantages. If the information is not used it will not be of an advantage.</td>
</tr>
</tbody>
</table>

Table 3. Summary of practices and problems from Knowledge Management
4.5 Communication Flows

When several persons are working in a team on the same site it can be good to have a single designated contact with the responsibility to disseminate formal information and knowledge to the co-workers and report status to the project leader. If no single contact exist it can be very hard to disseminate the same information to all team members and to get a consistent view of the team’s status. To have a designated contact is also an advantage when there is a new team member or when somebody returns to the team for a short time. In such cases there is seldom time to create a circle of contacts and an overall view of the product, so it is great if someone can take care of communication, join the meetings, and be of general support.

We had a person that took the role as team leader on one site. He kept an eye on what was going on and helped the others. He also managed some technical issues and made sure everything worked on that site, pulled some strings and so on. It worked great on that site. They were like a separate team inside the larger team and very independent.

To have a designated contact on each site also creates some disadvantages. There is a risk that the information is distorted by the contact: the shorter the information paths, the less risk of distortion. Some essential information can also be lost on its way to the receiver.

Over time, informal communication channels will be created and the task of the contact will in the end be to manage and disseminate primarily formal information.

Quality of communication

Since the communication often decreases when working in a distributed environment it is important that the quality of the communication is as good as possible. One way to increase the quality of communication and reduce the number of misunderstandings is to take minutes of meetings. This is often not enough. A way to further decrease misunderstanding is to send recurrent emails with status information, or to make regular calls only to say: hey, what’s going on?

Effects of reduced informal communication

Several other important effects from reduced informal communication were mentioned:

- It can be hard to know what is extra important right now and how important it is. This information is often circulated with informal communication.

- Misinterpretations of rules are often affecting a whole site.

- It takes time to disseminate information to remote sites.

- There is a risk that a feeling of being outsiders arise at the sites, with a lower motivation as the result.

These problems must be managed. The more dispersed the development is, the greater is the risk of problems. It is often good to gather the development to two larger sites instead of several small.

Group size

When the team at a site reaches a certain number of members a critical mass is reached, where there is a rich informal communication in the group and where the number of channels for com-
communication between sites let all necessary information pass. The work can then be performed
without great inconvenience over a distance. This is illustrated below:

If you are 20 persons, with 10 on each site, I don’t think it matters if you’re working from one or
two sites.

Tools for assisting communication

Those with extensive experience of distributed development consider phone calls as superior to
emails for communicating. It is hard to get a real dialogue with email, and waiting for answers
often creates a delay. An opinion is that there are too many emails and too few phone calls.
One reason for not using the phone is that you do not know the person on the other end. The
probability of calling somebody you know seems to be higher.

A good alternative to the phone is to use a tool that lets you both speak and share a workspace,
e.g. Microsoft NetMeeting. This gives you the possibility to get an even richer communication. The
advantage is illustrated below:

The phone is worthless compared to NetMeeting. The difference is that you can work with his
model… It’s simply that you can see and work on the same screen. We were working in the same
tool and coded together, like pair programming.

It is of great importance that all tools used to enable the distributed development works all the
time and are in place from start. If a tool does not work in the distributed environment it will
make the development on the remote sites much harder, or even impossible.

A summary of issues concerning communication can be seen in Table 4.

<table>
<thead>
<tr>
<th>Summary of Communication Flows</th>
</tr>
</thead>
</table>
| Use a single contact to disseminate information to remote sites and to get status inform-
|   ation.                        |
| Increase the quality of communica-
|   tion.                        |
| Try to reach a critical mass in group size at the sites. |
| Encourage usage of tools that can give a rich dialogue. |

Table 4. Summary of practices and problems from Communication Flows

4.6 Leadership

If communication fails in a distributed project it can result in terrible problems. One of the inter-
viewees gave an example of the possible result:

The worst case is that it doesn’t work at all. If it’s located at one site something will always work.
Everyone has a task, so if you are sitting pretty close to each other and attend the same meeting you
will get similar information, and you will always get something. But if the communication between
two sites does not work, you won’t get anything. Different interfaces are used, different ways of work-
ing, completely different functions are implemented etc. There’s no stop in how completely wrong it can
get.
To avoid this, the leader has to make sure communication channels are created. This is also an opportunity for the leader to establish exactly those communication channels between the sites that will make the work most efficient. This is rarely possible in co-located development where there always are lots of informal contacts already established. You often ask someone you know when you have questions, but it is not certain that this person has the correct information, even if he believes so. This can for example create different views of how a requirement shall be interpreted.

The leader cannot act in another way towards the employees at the remote site; the leadership must be the same as for the co-located employees. When a problem occurs on another site you cannot expect it to be solved by itself. The only right thing to do is the same as for the local leadership; act at once. If the developers are left aimless on drift they will have to make financial decisions that in reality are the responsibility of the leader. Even if the developers have a very good insight into their parts, they seldom have the complete picture of the project and dependencies to other interrelated project.

*If your way of leading works on Site X, you can be damn sure it will work great also on Site Y.*

Table 5 summarizes identified leadership issues.

<table>
<thead>
<tr>
<th>Summary of Leadership</th>
</tr>
</thead>
<tbody>
<tr>
<td>The leader is responsible for making sure that communication between the sites works.</td>
</tr>
<tr>
<td>The leader has the possibility to establish the most efficient communication channels. This opportunity is almost nonexistent in co-located development.</td>
</tr>
</tbody>
</table>

Table 5. Summary of practices and problems from Leadership

### 4.7 Quick Project Changes

It is often harder to motivate the individual employee to make a short temporarily step-in into a project than it is to get an agreement with managers and project leaders. The developer sees his current project as important, and by not working with it for a couple of weeks the timetable will be ruined and milestones missed. The motivation to change project can be extra hard at remote sites since the urgency and panic in the project does not show as clearly as at the main site.

*The work in the project was often strenuous and there was always a lot to do. You can feel it when you’re working here, but when they’re working at remote sites I don’t think they are noticing it in the same way. I think it gets a little harder, maybe you don’t realise the urgency. If you don’t, you won’t understand the reasons for priorities.*

A temporarily change does also put more strain on a developer at a remote site since the channels for communication to the main site that existed earlier might be more or less dead now. A solution to the motivation problem is for the leader to put additional efforts in making what is prioritized clear.

---

5 The distance between Site X and Site Y is approximately 150 km.
Some resources are probably needed for fixing errors after delivery. This makes it important not to plan all resources to another project directly after delivery, in order to avoid the problems described above.

**Vertical communication**

Vertical communication, i.e. communication between different hierarchical levels in the organization, is above all used when quick priorities must be set and when there are great problems. These channels of communication are consequently not used and maintained to the same degree as the horizontal channels. This can be troublesome since quick resolutions often must be found when problems arise. To manage this problem it can be useful to have priorities between projects. It is also important to have clear roles to make communication channels visible.

... it would be an advantage for everyone to know that this has priority X. You wouldn’t have to go through the organization for a day or two just to get hold of the project leader you need to discuss the prioritization with.

A summary of issues with quick changes and decisions can be seen in Table 6.

<table>
<thead>
<tr>
<th>Summary of Quick Project Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>It can be hard to motivate a co-worker to a temporary project change.</td>
</tr>
<tr>
<td>Vertical communication is almost only necessary when there are problems. It is hence important that it is prearranged so as to be efficient.</td>
</tr>
</tbody>
</table>

Table 6. Summary of practices and problems from Quick Project Changes

### 4.8 Product Ownership

The larger site often owns and has the responsibility for the entire product, also the parts developed by the smaller sites. A common reason is that there is no culture of sharing ownership and an attitude that the responsibility should be safeguarded in the interest of the main site. This attitude can create several problems:

**Motivation**

The motivation is often increased by owning what you are working with. The motivation can increase in the work in general, but it can also be easier to motivate co-workers to a temporarily project change.

Above all, I believe it would be possible to gain motivation from the feeling of “Wow, it’s ours!” You get a completely different verve and it is much easier to internally motivate why we should lift X from this work when we have problems. We will suddenly decrease the number of persons necessary to be agreed with.

**Product leading**

The different sites are eager to take responsibility for and further develop the product they have created. This is something they rarely get money for automatically. The product leaders responsible for the money on the main site do not have a good insight into the product developed on the remote sites, and because of this no greater interest in the further development. They are more motivated in developing the locally developed parts that they know well. This means that the sites are forced to ignore and pass over the product leaders. Something that can result in both losing
the overall picture and making the product leaders upset. The best solution is probably cooperation on equal terms, where the small sites get a larger responsibility than otherwise.

Table 7 summarizes issues concerning ownership and responsibility.

<table>
<thead>
<tr>
<th>Summary of Product Ownership</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation can increase if the remote sites have responsibility for the product they develop.</td>
</tr>
<tr>
<td>Both the responsible site and the developing sites should be involved in decisions on further development of products.</td>
</tr>
</tbody>
</table>

Table 7. Summary of practices and problems from Product Ownership

### 4.9 Methodological issues

When working with distributed development it is very important to use a development method. The use of a method is probably even more important for distributed development than for colocated development. It is also very important that all sites use the same method. The method creates a framework on which the communication is built, and it is of course crucial that all sites understand the communication framework used by other sites. The method specifies which artefacts that should be specified formally; artefacts which are the core in the development and must be interpreted by everyone in the same way.

It is important to be consequent in using the method. A quick ugly hack to fix something made at a time of momentary panic will only cause even greater problems later. One advantage of the decreased informal communication in distributed development is that this kind of panic rarely propagated to other sites, where they are following their method systematically. This is almost always rewarding in the long run.

The method helps in formalising the communication, but it is important not to rely only on the communication from the method. It is also important to talk with each other to increase understanding. A good way in achieving this is to include developers in the tasks prior to theirs, e.g. let the programmers get insight in the requirements specification. The effect of not doing this is illustrated below:

> It’s very important that the programmers take part in the design and maybe also the requirements specification, so they understand it. If you give a programmer a pile of 300 pages and say “Here are the requirements, now write the code!” he will probably read the first requirement and then say “Now I want to do something. Hey man, how do you want it to work?” He won’t look in the damn document. We have experienced this, that they do not have the energy to read requirement atom no. 763. Instead they’ve asked to try to get an understanding. … The result can be that it’s not the one who knows correct that decides how the system works, but instead the most powerful.

Some methodological techniques that can be useful in order to be more successful in distributed development have been mentioned in the interviews. They are discussed below:

**Iterative development**

To work in an iterative way helps reducing many problems in distributed development. The developers are forced to communicate at the start and end of each iteration, and in the integration at the end of the iteration you have a test that shows if the development ran in different direc-
Misunderstandings can easily bemuse the development, but the development won’t run for a long time before a mistake is noticed.

Since distributed development is more prone to misunderstandings it is important that the iterations are short. This is necessary both to keep a shorter time between integrations and to have less information to interpret at the start, and thus less to misunderstand. The drawback is that short iterations are more costly in administration. It is important to make sure that the part of time spent on administration is not too large. Irrespectively of the length of the iterations, what is most important is that they are well planned.

**Testing**

To further decrease misunderstandings tests should not only be performed at the end of iterations. As soon as there is anything to test, the testers should get the code. This will not only mean that errors are detected earlier, but also that the communication channels between developers and testers are improved. The communication will be continuous and at the same level during the whole iteration instead of as a large bulge at the end.

**Model based development**

Model based development can be a way of increasing the success in distributed development. With model based development more is described with visual models, which are easier to understand than source code for other developers. This makes it easier to communicate and share knowledge.

**Distributed pair programming**

Pair programming is said to be a technique that can decrease the number of software errors. It is possible to use pair programming also in a distributed environment. This is made possible by utilizing a tool similar to Microsoft NetMeeting, where both developers can share the same workspace. Distributed pair programming can possibly be even more efficient than co-located since knowledge from two sites is involved in the work. This helps eliminate misunderstandings and misinterpretations. Furthermore, working together is a good way of learning to know each other.

Table 8 shows a summary of issues concerning methods.

<table>
<thead>
<tr>
<th>Summary of Methodological issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is very important that a development method is used and that it is the same on all sites.</td>
</tr>
<tr>
<td>The communication enforced by the method is never enough.</td>
</tr>
<tr>
<td>Iterative development is a good way to increase communication and decrease misunderstandings.</td>
</tr>
<tr>
<td>Tests should be run all the time, not only during deliveries.</td>
</tr>
<tr>
<td>Model based development can be a way to increase understanding.</td>
</tr>
<tr>
<td>Distributed pair programming increases the knowledge involved in programming and can help decrease misunderstandings.</td>
</tr>
</tbody>
</table>

Table 8. Summary of practices and problems from Methodological issues

---

6 Model Based development is a development technique where the model is central for both design and implementation. The model is executable and contains all code.
5 Discussion

The most interesting finding in this study is that distributed development should not only be seen as a necessary obstacle, but also as something that can give additional strength to the development. The enabler most clearly pointed out in the interviews is the possibility for the leader to create an efficient network of communications between the developers. The most efficient network is probably hard to predict, but a good start can be to look at dependencies between roles and handovers specified by the development method. Another interesting way to gain surplus values is by utilizing distributed pair programming, where it is possible to make use of knowledge from two sites. It can make practitioners realize that distributed development isn’t necessarily a bad thing, instead it can be a way to gain advantages.

There are not lots of existing research with focus on how to gain surplus values from distributed development. Most of the available research looks on how to manage the problems involved in working in a distributed environment. This is of course important as well, but reveals only one side of distributed development. Orlikowski (2002) supports the idea of using distributed development as a means to gain surplus values. Orlikowski sets the main focus on organizational values that help creating a sound environment for distributed development, not on the specific practices that can create the advantages.

This study has also identified practices that can help overcome some of the problems involved in distributed development: the importance of meeting face-to-face, knowledge management as a means of disseminating knowledge, the importance of having well established channels for communication, the important role of the leader, that all sites need responsibility, the importance of using one method etc. Several of the identified practices can be found in the literature review in chapter 3. Herbsleb & Grinter (1999) and Pyysäinen (2003) point out the importance of meeting face-to-face, Kraut & Streeter (1995) describe the importance of an efficient and qualitative communication, Paasivaara & Lassenius (2004) and Bass & Paulish (2004) support the use of iterations as a way to be more successful. A complete survey of which of the practices that has already been identified in previous research is however outside the scope of this study. What is most important is the fact most of the practices point at, that it requires additional effort to manage distributed development. A laissez-faire attitude might work in a co-located environment, but in a distributed environment this attitude will most likely wreck the project.

Some of the identified practices have traits similar to Open Source development: Knowledge sharing is very similar to both what Augustin et al. (2002), Persson et al. (2005); Crowston, (2002) say about Open Source and what Orlikowski (2002) has found in her study. The idea of shared ownership and responsibility is similar to the Open Source ideas described by Fuggetta (2003).

An estimate on the ability of current software development methods to support distributed development could not be produced. It is however important to remember that what is most important is that a method is used, and that it is the same method on all sites, as this study has shown. What method is used is of less importance. It is still unwise to downplay the significance of the chosen method, especially since several methodological practices which can create a more successful project has been found. As have been shown in this study, it can be hard to get the needed methodological knowledge from developers. To gain real methodological insight it would be more suitable to interview method experts.
In the study I have avoided looking on how specific tools can help with distributed development. In spite of this, some recommendations concerning tools have been presented. The reason for this is that the interviewees mentioned the tools as a means of being successful.

Two intra-organizational cases with all sites within the same nation have been studied. Many of the identified practices can probably be generalized outside of this specific domain, but that is for a future study to confirm. Outside of this domain there are probably some additional problems as well, e.g. cultural differences, different languages etc.

Since the practices have been identified by using a qualitative approach with only five interviewees, it is important to make further studies to either confirm or restrict the validity of the identified practices. Using a framework to measure success could have strengthened the reliability of the results, but that would have required a much larger study. Several of the identified practices are furthermore quite subjective in nature and hard to quantify. A falsification might be hard to achieve since the practices might work in some organizations, but the generalizability might be restricted by other findings.

As was mentioned in section 2.4.4 it is important to elaborate on future ethical consequences of the results. It is hard to believe that the results presented here should lead to inhuman ways of working; the presented practices will rather lead to a more humane way of working, with more personal meetings.
6 Conclusion

The main objective of this study was to identify practices that can make geographically distributed development more successful. Such practices have been identified in two categories: practices that manage the problems of distributed development, and practices creating surplus values in distributed development. The latter is especially interesting since most of the existing research mainly focus on managing the problems associated with distributed development. The most important practice creating surplus values is the possibility for the leader to create the highly efficient channels for communication between sites. Some of the important practices for managing problems specific to distributed development are: learn to know the other developers by meeting, disseminate project knowledge, and share product ownership.

Several intermediate objectives were used to reach the main objective and to give the study structure and focus:

Establish a definition of geographically distributed development and identify differences to co-located development. This has been done by conducting an extensive literature review, covering problems with work in a distributed environment and proposing solutions on how to successfully carry out a distributed project. The literature review has been used as a basis for creating an interview guide.

Study how the problem is managed in different organizations. This objective has been the most important objective of the study. Two different cases were studied to gain insight in how organizations manage the problems with distributed development. A number of interesting work practices and recommendations were identified in this part of the study.

Examine how suitable current development methods are for distributed development. This objective has not been fully realized. The idea was that the interviewees from the studied organizations should identify how suitable their methods are for distributed development. Even though the studied organizations were mature in working with development methods, the interviewees could not give enough answers on how well the methods support distributed development. Several methodological practices have been identified as successful, so an intermediate conclusion can be drawn from the objective by matching identified practices and problems to development methods. What is most important according to a majority of the interviewees is not which method that is used, but that a method is used and that it is the same on all sites.

6.1 Future research

It would be interesting to make a study with the aim to identify more practices that helps in creating surplus values from distributed development. A good starting point is to looking at the values identified by Orlikowski (2002). The Open Source community is successful in using distributed development in many projects. Work practices from Open Source might be another starting point for finding such practices. The literature review in this thesis includes a discussion on the issue of motivation but there are probably other important factors as well.

Similar studies might also be interesting to do in different environments to see if the results can be transmitted to other domains. Orlikowski (2002) shows that distributed development can cre-
ate surplus values also in international projects, although it is not certain that it is the same prac-
tices that are successful in that domain.

One of the objectives of this study was to see how well different software development methods
can fulfil the special demands in distributed development. Since that objective could not be ful-
filled it would be interesting to make further studies in that area. The practices and problems
identified in this study can be used as a basis for what must be demanded from the methods.
References


Appendix A. Interview Guide

Organization

How are you organized in the project?
Which part of the project does you unit have?
How well does the project structure fit the organization structure?
Do you think that the organizational structure makes the work more efficient?
Who has the greatest power over decisions?

Decision power

How does it affect you?
What is managed with informal communication?
Which persons did you have contact with last week?
Which issues were dealt with?
What was the environment for the contact?
What is the division of communication between you own site and cooperating sites?
How is formal communication used in the project?
How much?
How often do you meet each other?
What is typically the reason?
Does it cause great inconvenience?
Advantage/disadvantage?
Division of vertical and horizontal?
What issues are managed with each?
Differences between external/internal?
Is there enough communication?
Is it inconvenient to communicate with other sites?
What communication do you miss?
How is it possible to communicate better?

Interview Questions

The project

In which ways is it distributed?
Which are the sites?
Project type?
Organization?
Interdependencies between sites?
How is the distribution experienced?
What are the effects of working distributed?
Positive/negative?

Ways of working

What is the best way of working in a distributed environment?
Which are the problems?
How can you avoid them?

Organizational culture

Describe the developers attitude towards working in a distributed environment.

Which are the fundamental values of the organization?
How do they affect the distributed work?
How does the motivation affect the distributed work?

Do you use a specific method?
What method?
Describe briefly
Does the method give support to distributed work?
Which parts give the most help?
What support for distributed work is missing?
Can you use the method as it originally was designed?
What have been changed?
In what ways does the method help in communicating?
Is the method a support or a limitation?
Is the same method used at all sites?
What is the effect?
Do you work iteratively?
What is the effect on the distributed work?

Does the method give support to distributed work?
Which parts give the most help?
What support for distributed work is missing?
Can you use the method as it originally was designed?
What have been changed?
In what ways does the method help in communicating?
Is the method a support or a limitation?
Is the same method used at all sites?
What is the effect?
Do you work iteratively?
What is the effect on the distributed work?

Quality

Describe the developers attitude towards working in a distributed environment.