SCIENCE • TECHNOLOGY • RESEARCH HIGHLIGHTS • VISIONS

Dissertation
80

Customer communication in distributed agile software development

Mikko Korkala

VTT

VTT SCIENCE 80

Customer communication in distributed agile software development

Mikko Korkala

VTT Technical Research Centre of Finland Ltd

Thesis for the degree of Doctor of Philosophy to be presented with due permission for public examination and criticism in IT116, at University of Oulu, on the April 10th at 12:15.

ISBN 978-951-38-8230-3 (Soft back ed.)
ISBN 978-951-38-8231-0 (URL: http://www.vtt.fi/publications/index.jsp)

VTT Science 80

ISSN-L 2242-119X
ISSN 2242-119X (Print)
ISSN 2242-1203 (Online)

JULKAISIJA – UTGIVARE – PUBLISHER

Teknologian tutkimuskeskus VTT Oy
PL 1000 (Tekniikantie 4 A, Espoo)
02044 VTT
Puh. 020 722 111, faksi 020 722 7001

Teknologiska forskningscentralen VTT Ab
PB 1000 (Teknikvägen 4 A, Esbo)
FI-02044 VTT
Tfn +358 20 722 111, telefax +358 20 722 7001

VTT Technical Research Centre of Finland Ltd
P.O. Box 1000 (Tekniikantie 4 A, Espoo)
FI-02044 VTT, Finland
Tel. +358 20 722 111, fax +358 20 722 7001

Grano Oy, Kuopio 2015

Preface

The journey I began in 2004 has been travelled by taking small steps. Sometimes very small steps. On occasion I stopped altogether for a while, lay down and watched the clouds sail through the skies. While I was lying there, I thought about just leaving everything behind, going home and doing something else with my life. I was tired of dragging myself over a rugged terrain towards a seemingly unreachable goal; a goal that I wasn't even sure existed. But lying on your back can be pretty boring so, time after time, I got up and thought about dragging myself just a little bit further along the road and up to the next milepost. When I reached these markers I felt the immense joy of discovery and feelings of accomplishment. It was these feelings that kept me going and made me resist the urge to turn around. I also really, really wanted to see what was there at the end of the road. Ultimately, my long march has led me to this final milestone of writing these lines. It is here that this journey ends, which makes me both happy and sad. Would I do this again, if given the opportunity? Yes I would. What would I do differently if I were to do it all again? Everything. You see, the world is full of exciting roads to travel and milestones to reach and who knows what miracles your adventure will reveal. Adventures are never easy, but one does not embark on adventures because they are easy, but because they are hard. This alone makes it worthwhile to begin another one. However, this time I will ride a bicycle.

There are several people that I have worked with during this project who deserve my sincere gratitude. First, I would like to thank my supervisor, Professor Samuli Saukkonen from the Department of Information Processing Science, University of Oulu, for his support during this work. I would also like to thank him for encouraging me to take this path in the first place, back then when I was still working with my Master's thesis. I also owe a lot to Professor Pekka Abrahamsson, my other supervisor. He has been my mentor and a source of inspiration throughout this effort. I would also like to thank Professor Frank Maurer from University of Calgary, Canada for the opportunity to work in his talented team from December 2010 to December 2011. This visit was an invaluable experience during which I learned a lot about how to do research.

I am also very happy that I had two very distinguished scientists as the reviewers of this thesis. Therefore, I would like to thank Professor Brian Fitzgerald from LERO – The Irish Software Engineering Research Centre and Professor Tore Dybå from

SINTEF, Norway for their excellent feedback about my work. I also wish to thank all the co-authors who have contributed to the articles.

I am also privileged since I have had so many talented and wonderful colleagues during my career. Since you are so many, it would be impossible for me to remember to name you all in here. Therefore, I won't do that and I know you will understand. However, a very special *thank you* goes to VTT Oulu's *"Lunch Club"* for all the good and less good times we experienced together. My friends at VTT Espoo deserve a *thank you* as well. I am very proud that I was able to be a member of our highly exclusive *"Cucumber Club"*. Wherever your lives take you, I wish you all the very best!

Naturally, I would like to thank my mother, my brother and my sister for their support. My deepest gratitude goes to Minna, my long time companion, soul mate, mother of our daughter Aino Sofie, and a friend. I thank you for all the years we have had together and I cannot wait for all those wonderful things that we have in front of us still to happen! I am also forever grateful for my daughter, since being a father is in my world the single most precious blessing a man can ever have.

Perhaps the happiest person there was when I told about my decision to go for a PhD was my father. Sadly, he is not here to see it happen.

I dedicate this work to my father.

Mikko Korkala.
Helsinki, Finland, 28th November 2014.

Academic dissertation

Supervisors

Professor Samuli Saukkonen
University of Oulu
Department of Information Processing Science
P.O. Box 3000, 90014 University of Oulu, Finland

Professor Pekka Abrahamsson
Faculty of Computer Science
Free University of Bozen-Bolzano
Dominikanerplatz 3 – piazza Domenicani 3
39100 Bozen-Bolzano, Italy

Reviewers

Professor Brian Fitzgerald
LERO – The Irish Software Engineering Research Centre
Tierney Building, University of Limerick
Ireland

Professor Tore Dybå
SINTEF ICT
Strindvägen 4
Trondheim, Norway

Opponent

Professor Markku Tukiainen
School of Computing
Joensuu Campus
University of Eastern Finland
P.O.Box 111
FI-80101 Joensuu

List of publications

This thesis is based on the following original publications which are referred to in the text as I–V. The publications are reproduced with kind permission from the publishers.

I Korkala, M. & Abrahamsson, P. 2004. Extreme Programming: Reassessing the Requirements Management Process for an Offsite Customer. In: the proceedings of the 11th European Conference on Software Process Improvement (EuroSPI 2004). November 10–12, 2004. Trondheim, Norway. pp. 12–22.

II Korkala, M., Abrahamsson, P. & Kyllönen, P. 2006. A Case Study on the Impact of Customer Communication on Defects in Agile Software Development. In: the proceedings of AGILE 2006 Conference. July 23–28, 2006. Minneapolis, MN, USA. pp. 76–88.

III Korkala, M. & Abrahamsson, P. 2007. Communication in Distributed Agile Development: A Case Study. In: the Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007), August 28–31, 2007. Lübeck, Germany. pp. 203–210.

IV Korkala, M., Pikkarainen, M. & Conboy, K. 2010. Combining Agile and Traditional: Customer Communication in Distributed Environment. In: *Agility Across Time and Space – Implementing Agile Methods in Global Software Projects*, eds. D. Šmite, N.B. Moe & P. J. Åkerfalk. Springer. pp. 201–216.

V Korkala, M. & Maurer, F. 2014. Waste Identification as the Means for Improving Communication in Globally Distributed Agile Software Development. Journal of Systems and Software. Volume 95, September 2014, pp. 122–140.

Author's contributions

Paper I: The first author was responsible for planning the case, collecting and analysing the data and writing the paper. The second author provided comments about the manuscript and supported the research process by providing feedback.

Paper II: The first author was responsible for planning the case, collecting and analysing the data and writing the paper. The second author provided comments about the manuscript and supported the research process by providing feedback. The third author contributed to the case by collecting defect-related data from one of the case projects.

Paper III: The first author was responsible for planning the case, collecting and analysing the data and writing the paper. The second author provided comments about the manuscript and supported the research process by providing feedback.

Paper IV: The first author was responsible for planning the case, analysing the data and writing the paper. The second author was responsible for collecting the data since the first author was not able to participate in this process. The data collection (i.e. the interview) was conducted in Ireland. The second author also provided comments about the manuscript and supported the data analysis process. The third author provided comments about the manuscript.

Paper V: The first author was responsible for planning the case, collecting and analysing the data and writing the paper. The second author provided comments about the manuscript and supported the research process by providing feedback and helping to reflect on the findings during the study.

Contents

List of abbreviations

AR	Action Research
ASD	Adaptive Software Development
DSD	Distributed Software Development
DSDM	Dynamic Systems Development Method
DXP	Distributed Extreme Programming
FDD	Feature-Driven Development
GSD	Global Software Development
IID	Iterative and Incremental Development
ISD	Information Systems Development
MRT	Media Richness Theory
MST	Media Synchronicity Theory
RUP	Rational Unified Process
XP	Extreme Programming

1. Introduction

In 1968 Edsker W. Dijkstra wrote an article titled "*Go to statement considered harmful"* in which he criticized the use of the "go to" programming statement. Dijkstra claimed that the command increases the complexity of the program code and makes it more difficult to comprehend and, hence, it should not be used (Dijkstra 1968). Since this early recommendation for improving software development, a plethora of different development approaches have emerged. This evolution has led to the emergence of agile development methods. Instead of trying to identify a complete set of requirements before the project begins and to eliminate as much change as early as possible, agile methods embrace change by accepting and incorporating it instead of eliminating it. The conventional beliefs relating to the ability to identify all the potential aspects, such as requirements and schedules, in the beginning of a project have been criticised e.g. in (Ramasubbu & Balan 2009, Larman & Basili 2003, Poppendieck & Poppendieck 2003); uncertainty can be seen more as a rule than an exception, and understanding this rule is paramount for companies if they want to succeed (De Meyer, Loch & Pich 2002).

Communication is important in software development despite the chosen development approach (Saeki 1995, Beck 2000, Bostrom & Thomas 1983, Edstrom 1997), and its role is further emphasised in agile methods. Instead of relying on communication through specifications, documentation and other impersonal artefacts, agile methods propose close collaboration between all those who are involved in the development effort and encourage them to interact, preferably in a face-to-face manner (Beck 2000). The customer is an integral part of an agile team, responsible for steering the project in terms of its contents. The agile customer is actively involved in the development process, providing the team with a set of product requirements and supporting it in terms of providing feedback about implemented features. This active steering aims towards implementing a product that serves its intended goals in the best possible way. (Beck 2000) Agile software development projects rely on deliberately vaguely defined requirements. For example Paasivaara and Lassenius (2003) claimed that software projects developing genuinely novel products face challenges regarding software requirements. In projects such as these, collaborating parties cannot hope to receive clear product specifications at the beginning of the effort (Paasivaara & Lassenius 2003). Hence, close communication and collaboration is essential throughout the project,

since the participants are continuously trying to understand what to build at the same time as they are developing the project. (Paasivaara & Lassenius 2003)

Developing software in a globally distributed fashion can be seen almost as a business necessity for companies (Damian & Moitra 2006). Agile development methods were developed to serve the needs of small collocated efforts that could enjoy the benefits of having a constant customer presence available. The role of the agile customer can be seen as quite demanding even without distribution, and taking a distributed development approach in the context of agile development is even discouraged (Šmite et al. 2010a). Distributed development creates a host of challenges for the companies, and communication is one of them. Noll et al. (2010) identified that temporal, geographical and cultural distances are the main barriers hindering global software development, and that these barriers further introduce challenges for effective communication. However, it can be difficult to have active customer involvement and communication in distributed settings, and these challenges also affect customer communication. In order to meet the challenges, several different solution proposals have been provided.

Attempts can be made to overcome the challenges resulting from geographical distance by using various communication tools. In the context of agile development, that emphasises interactive and informal communication, solutions such as videoconferencing are proposed to be applied (Kircher et al. 2001, Sureshchandra & Shrinivasavadhani 2008). Regarding temporal distance, overlapping work hours can be utilized but this can be very consuming and may lead to overtime work (Holmström et al. 2006a, Sarker & Sahay 2004, Conchúir et al. 2009). Applying communication policies to reduce communication delays caused by temporal distribution could help with this particular distance (Vax & Michaud 2008). Cultural distance can be mitigated by involving people who understand all the languages involved (Layman et al. 2006), by rotating people across the sites (Ebert & De Neve 2001) and by using experienced domain experts able to make the issues transparent (Summers 2008). Given the large number of challenges, the question of improving customer communication in distributed agile software development becomes relevant.

1.1 Research questions

This section presents the research questions of this thesis. The main research question of this work is further elaborated by three additional sub-questions, each targeting a specific area of interest that contributes to the main research problem.

For the purposes of this study, a clarification of the central terminology is necessary. In this work, customer is defined as “the source of information and feedback for successful implementation of a software product”. This definition is derived from (Pikkarainen et al. 2011). In addition, distributed software development is defined as a development approach in which the stakeholders involved in a development effort are separated by a geographical distance ranging between the cross-town scenario (Prikladnicki et al. 2003) in which the stakeholders are dis-

tributed within a same city to the continental scenario (Prikladnicki et al. 2003) in which the stakeholders are distributed across continents.

In order to identify the central aspects that are required for improving customer communication in distributed agile software development, the main research question of this work is formulated as follows:

RQ: How is it possible to improve customer communication in distributed agile software development?

In order to provide the answer to this problem, the following sub-questions are defined that approach the problem space from different perspectives.

Communication is important in every software development approach, but it is especially emphasised in agile methods that emphasize face-to-face communication. Especially in distributed development, efforts to achieve such communication can be very challenging, and hence the effects of not enjoying active customer participation and therefore communication following the propositions made in agile methods should be understood. Therefore, the first sub-question is defined as:

Q1: Why is customer communication important in distributed agile software development?

The close collaboration between the stakeholders is emphasised in agile methods. In distributed settings, achieving this close relationship can be challenging and hence the mechanisms for involving the customer in the development process need to be studied. Sub-question 2 aims to provide insights into this aspect.

Q2: How is it possible to involve the customer in the development process in distributed agile development in order to ensure communication and feedback?

Agile methods focus on constant improvement of the development process. In order to follow this principle, it should be understood what means, practices and tools are required to improve customer communication so that it provides the necessary information in order to complete the development project successfully. In order to understand this aspect, the final sub-question is posed as follows:

Q3: What are the means, practices and tools for improving customer communication in distributed agile development?

1.2 Scope of the research

Agile methods have entered the world of distributed development, which appears to be a very important development approach for companies. Therefore, distributed development context is one of the aspects of this thesis. Considering the importance of the customer in agile development and the challenges it faces due to

distribution, customer involvement is one of the viewpoints taken in this study. Another central aspect of agile development is constant improvement of the development process (Agile Manifesto 2001). Whereas there are existing recommendations about how to ensure and improve communication in distributed agile software development, due to the importance of constant reflection in agile methods this aspect is included in this work. Figure 1 describes the scope of the study, which is further illustrated in the intersection of the three aspects of this thesis. The scope appears as the darkened section of the figure.

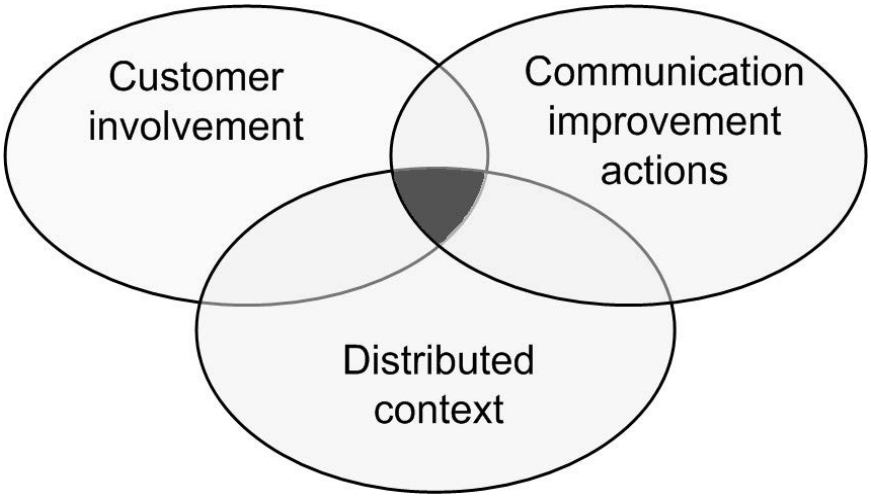

Figure 1. The scope of the thesis.

1.3 Organization of the thesis

The rest of this work consists of the following sections. Section 2 focuses on the theoretical background of the study, discussing terminology and providing a brief history of method development ranging from ad hoc methods to agile methods. In addition, Section 2 discusses agile software development in general and in distributed contexts in more detail.

Section 3 introduces the aim of communication, the elements and actors involved in it, the differences between formal and informal communication, communication in agile software development and the role of the customer in agile software development. In addition, challenges and their solution proposals identified in the previous literature are discussed and the efficiencies of communication media are examined from theoretical perspectives. Section 4 describes the research design, and section 5 presents the individual contributions of the research articles and summarizes the results. In Section 6, the key implications of this work for both theory and practice are discussed. Section 7 concludes the work. The original articles are included at the end of this work.

2. Background of the study

The aim of this section is to provide an overview of the main topics of the study. At first the terms software process, software process models and the distinction between methods and methodologies are discussed. Section 2.2 explores the history of software development methods, and agile software development methods are discussed in Section 2.3. Section 2.4 discusses distributed software development in general, with particular reference to distributed agile software development. In addition, the current state and the future of agile development methods are discussed. Finally, Section 2.6 provides a summary of the topics discussed in this section.

2.1 Software process, software process models and methods

According to Humphrey (1995, p.4), software process can be defined as "the sequence of steps required to develop or maintain software". In addition, software processes aim towards providing the "technical and management framework for applying methods, tools and people to the software task (Humphrey 1995, p.5). Further, Sommerville (1996, p. 269) claimed that software process "consists of the activities and associated information that are required to develop a software system". Boehm (1988, p. 61) stated that "the primary functions of a software process model are to determine the order of the stages involved in software development and evolution and to establish the transition criteria for progressing from one stage to the next". Software process models thus provide guidelines considering the order in which the major tasks of a project should be carried out and address the following questions as defined by Boehm (1988):

1) What is done next and 2) for how long it will be done.

Software development methodologies on the other hand focus on how to navigate through these stages and how to present different phase products (Boehm 1988). However, Conboy (2009) pointed out the vague definitions of both *method* and *methodology*. In his study, Conboy (2009, p.329) reached the following definition of a method in the context of Information Systems Development (ISD): "An ISD

method encompasses the complete range of practices involved in the process of designing, building, implementing, and maintaining an information system, how these activities are accomplished and managed, the sequence and frequency of these activities, as well as the values and goals of all of the above". However, Conboy (2009) claimed that over the years the notion of a method more as an idea and an enactment has been accepted in the field of ISD. Conboy (2009) used the term "method" while discussing agility and agile approaches and a similar interpretation is also adopted in this thesis.

2.2 From "code-and-fix" to iterative development

In this section, the history of software development process models is described briefly. In addition, traditional software development is described in more detail. In this dissertation, traditional software development is synonymous with plan-driven development. Plan-driven software development is a software development approach in which the software is developed following specific processes, commencing with the stage of gathering requirements and ending with the final code (Boehm & Turner 2003b).

According to Avison and Fitzgerald (2003) the computer applications of the 1960s and 1970s were implemented without formalized or explicit development approaches and the emphasis was on programming and solving technical problems. Boehm (1988), referred to this approach as the "code-and- fix" approach, which had two steps; the code was written and the problems were then fixed. The problems related to this approach led to recognizing the needs for improving the development process, suggesting that separate design, requirements identification and as well as test and evolution planning phases along with other tasks related to preparation were needed prior to coding (Boehm 1988). Boehm's (1988) brief discussion continues with a *stagewise* model. In his work, Boehm (1988) referred to the development of the SAGE system, which had led to recognising certain problems and to the development of the stagewise model in order to address these issues as early as 1956. Benington (1983) categorized these problems as follows: *1) Computer operation, 2) Program or system reliability, 3) Supporting programs* and *4) Documentation*. According to Benington (1983, p.299), the problems emerged from the size of the computer programs; they "were too large for one person to grasp entirely". In order to tackle these problems, Benington (1983, p.306) suggested that the software should be implemented in successive stages as follows: *operational plan, operational specifications, program specifications, coding specifications, coding, parameter testing, assembly testing, shakedown* and *evaluation*.

Traditional plan-driven software development methods are described as document-driven, code-driven and traditional methods (Boehm 1988). Plan-driven software development is an engineering approach in which the software is developed following specific processes, commencing with the stage of gathering requirements and ending with the final code (Boehm & Turner 2003b). There are

several methodological approaches and models describing how to develop software in a plan-driven fashion (Boehm & Turner 2003b).

In this work, plan-driven methods are referred to as traditional methods. These traditional methods approach software development from the viewpoint that systems are fully predictable and specifiable and that they can be built through careful and extensive planning (Nerur et al. 2005). One of the most well-known traditional approaches is the Waterfall model introduced by Royce (1970). As Boehm explained in (1988), the Waterfall model introduced two major enhancements to the stagewise model; 1) the introduction of feedback loops between stages and the guideline to limit the loops to successive stages in order to minimize costly rework resulting from feedback across several stages, and 2) a piloting approach in which the first version is a pilot version of the software and the second one is the final product, which should be implemented based on the feedback from the pilot version. The software is thus built twice and the first result simulates the final product. (Royce 1970). The Waterfall model is depicted based on (Royce 1970) in Figure 2.

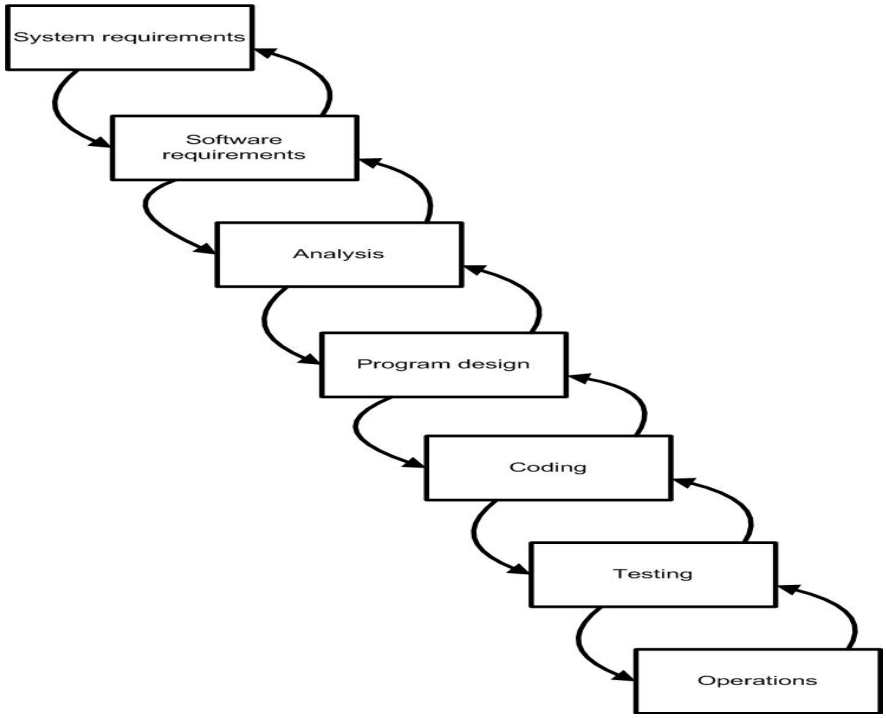

Figure 2. The Waterfall model as depicted by Royce (1970).

Even though there are hints of iterativity, feedback and adaptivity present in the Waterfall model, the iterative step based on feedback has been lost in most descriptions of the model. The Waterfall model is often misunderstood as a single-

pass approach, to which the approach devolved with its strict sequence of requirements analysis, design, and development phases (Larman & Basili 2003). Nevertheless, the Waterfall approach has helped to resolve many of the previously encountered difficulties in software projects and it has established itself as the basis of most industrial and governmental software acquisition standards (Boehm 1988). According to Boehm and Turner (2003b) the strength of traditional plan-driven methods lies in the comparability and repeatability that standardization brings.

However, even with refinements and revisions, such as extensions supporting incremental development and evolutionary change, the Waterfall model has encountered several fundamental difficulties. Boehm (1988) stated that the primary source of difficulties of the Waterfall model has been its emphasis on using highly detailed documentation as the completion criteria for early design and requirements phases. For example, document-driven standards have resulted in many projects writing elaborate specifications of user interfaces and decision-support functions that are understood poorly, which in turn have resulted in defective designs and the development of large amounts of unusable program code. (Boehm 1988) Benington (1983, p.301) endorsed this opinion by stating that aiming for a precise understanding of what the software needs to do before a single line of code is produced "can be terribly misleading and dangerous".

In order to further support the drawbacks of the Waterfall approach, Larman and Basili (2003) referred to the Standish Group's 1998 report "CHAOS: Charting the Seas of Information Technology", which charted 23 000 development projects in order to identify the failure factors. Larman and Basili (2003) concluded that according to the report, the top reasons for project failure were related to Waterfall practices. In addition, for example Avison and Fitzgerald (2003) discussed the caveats of the Waterfall approach, stating that it had problems related to e.g. meeting the real business needs, user satisfaction and with modelling processes that are unstable due to changes both in the market and in the business. Further, Parnas and Clements (1986) stated that the development approaches that can be classified as top-down approaches result from a desire to have a systematic and rational way of designing software. Further, they listed the following reasons why design processes that are claimed to be rational and systematic will remain as idealizations.

1. In most cases the users of the system do not know what they want from it and cannot explain what they know.

2. Even though the requirements are known there are still many other facts that need to be known while designing software. Some of the details will reveal themselves only during the implementation process and this learning process may invalidate the design.

3. Although all the relevant facts are known prior to development, humans are unable to fully master the plethora of details that need consideration while designing and building a correct system.

4. Even if all the needed details could be mastered, external factors will cause changes in all but trivial projects.

5. Human errors are inevitable no matter how well the relevant facts are collected and organized and how rational the design process is.

6. Some preconceived ideas are tested in the project despite the fact that they are not derived from the requirements by a rational process, but emerge from other sources.

7. For economic reasons, the use of software developed for some other project is often encouraged, or the software being implemented is encouraged to be shared with another ongoing project. The resulting software is often a compromise, and does not represent the ideal software for both projects.

For these reasons, Parnas and Clements (1986) claimed that creating an error-free design from requirements in a rational fashion is unrealistic. In fact, Parnas and Clements (1986) presented a strong critique of publications reporting a rational design process being applied in even small program developments. They claimed that the results had been "polished" to describe the process in the way that the author would have liked it to go instead of reporting what actually happened. Parnas and Clements (1986, p.252) introduced a concept of "faking the rational design process", which suggests that if a rational process cannot be followed, it can still be followed as closely as possible if the documentation that would have resulted from following the ideal process is written. Parnas and Clements (1986), in fact, promoted this "faking" process in terms of communication, since this rationalized documentation provides the reader with what he needs and results in a product that can be understood, reused and maintained.

Despite the challenges and the criticism, the Waterfall model is still widely used in the industry and has its benefits, such as predictability and focus on detailed planning of both the architecture and the structure of the software, which are both essential attributes when dealing with large systems (Petersen et al. 2009). Further, Petersen et al. claimed in (2009) that many of the reported problems related to the Waterfall model are based on experiences and beliefs instead of empirical findings. To address this knowledge gap, they conducted a case study concluding, perhaps contradictory to their claim of the benefits of the Waterfall model, that the

issues identified in the literature were found in their work and that the Waterfall model is not suitable for large-scale development (Petersen et al. 2009).

Boehm (1988) continued his discussion of the evolution of software development approaches with *Evolutionary Development Model*, which is a concept introduced by Gilb (1981). In this approach, development consists of increments of working software that expand as a result of the experiences gained from its functionality. However, according to Gilb (1981), the main idea behind the approach is to speed up development by releasing these increments of the system at an early stage. For example Royce (1990) discussed the benefits of the evolutionary approach by stating that it provides a systematic way of identifying quality problems early in the development. However, evolutionary development is not free from problems. According to Boehm, (1988) it was difficult to distinguish the evolutionary model from the code-and-fix approach and the model expects, often unrealistically, that the flexibility of the user's operational system will adapt to unplanned evolutionary paths. In these cases, the stages are often pursued in the wrong order. In order to provide an example, Boehm (1988) stated that a significant amount of code, which is hard to change, is written before taking long-range architectural and usage considerations into account.

The *Transform model* followed the evolutionary approach. The transform model is described in (Balzer et al. 1983) and presents an iterative development approach taking into account product adjustment based on changing needs. In addition, this model has a strong focus on end user involvement in the process. In this approach, the changes are made in the specifications themselves by the end user. This approach should drastically simplify the maintenance problem. "The maintenance problem" discussed by Balzer et al. (1983) is twofold. First, it results from the lack of technology to manage the often undocumented, informal and labour-intensive activities that constitute software development processes. The second aspect is the fact that maintenance is made at the source code level, in which the important information is dispersed. This latter problem makes the software difficult to understand. (Balzer et al. 1983). As the difficulties associated with the transform model arise from its emphasis on automation, Boehm (1988) proposed that the model is suitable for small applications in limited application areas, and that it suffers from difficulties similar to those associated with the evolutionary approach. As an example of the latter, Boehm (1988) claimed that similarly to the evolutionary approach, the transformation model expects that the flexibility of the user's operational system will adapt to unplanned evolutionary paths.

The *Spiral model* (Boehm 1988) is an iterative and incremental development approach. Each cycle of the spiral has typically the following steps:

1. Identification of 1) objectives for the proportion of the product to be elaborated, 2) alternatives for implementing this proportion and 3) the constraints related to application of these alternatives, such as costs and schedule.

2. Evaluation of alternatives related to constraints and objectives. In addition, this step identifies those areas that are uncertain and may represent a risk to the project. These risks are further identified and resolved.

3. Development of the features and their verification.

4. Planning of the next phase of the product.

Reviewing the results of the cycle plays an important role in the Spiral model. This review includes all the products implemented during the cycle, including the plans for the next one. This review includes the relevant stakeholders that are concerned with the product. The major objective of the review is to ensure mutual commitment of all stakeholders considering the approach for the next phase.

In addition, the Spiral model is a risk-driven development approach. Each development cycle begins with risk analysis. The Spiral model is not rigid in the selection of different development approaches, since based on the risks, any mix of different development approaches (e.g. prototyping and simulation) can be applied for achieving the goal of the given cycle. (Boehm 1988). There are downsides related to this approach. For example Wolff (1989) reported such findings from a study focusing on the implementation of a management information system. Whereas the study reported generally positive results concerning the Spiral model, it was also reported that passing the long term objectives and plans from cycle to cycle should be defined more clearly, along with their integration with the shorter term plans and objectives (Wolff 1989). In addition, Wolff (1989) saw the different activities of the cycles, such as defining the objectives and identifying risks, as planning activities and continued that there is a need to define more clearly how any planning carried out following the model relates to the model itself, since the model can be seen as a “skeleton plan” for the project itself.

The evolution of software process models has moved towards more and more change-driven approaches to software development. As a culmination of this trend towards change-driven models, agile development was introduced in the late 1990s. Agile software development will be discussed in the next section.

2.3 Agile software development

According to Åkerfalk and Fitzgerald (2006), despite decades of software development, many of the characteristics of the “software crisis” are still prevalent in the industry. The defined symptoms of this phenomenon are exceeded budgets and schedules along with poor quality of the end product (Glass 2006). As stated in (Ågerfalk & Fitzgerald 2006), agile methods have been proposed as a paradigm that potentially addresses these problems by promoting flexibility, teamwork, innovation and communication. Further, Turk et al. (2005) claimed that agile methods emerged primarily to support the development of high quality software meeting the

customers' needs in a timely and economical manner. These reasons are now almost a decade old, and perhaps something has changed in why agile methods are being adopted by companies. A company named VersionOne[1] conducts annual surveys of agile development. In their latest 7th annual survey[2] with a total of 4048 respondents from companies having a median of 100 employees, the top five most important reasons for adopting agile developments methods were as follows: accelerated time to market was the most important reason, with 30% of respondents claiming it to be very important. The second most important aspect was managing changing priorities with a score of 29%, followed by better alignment of business and IT (23%). The fourth most important factor was to increase productivity (18%), while the fifth reason, improved quality, gathered 17% of the "highest importance" votes. However, reducing costs, which is one of the key "original" targets of agile development methods, gained a share of only 10%.

Abrahamsson et al. (2002) stated that it is widely acknowledged that the starting point for various different agile methods was the emergence of Extreme Programming (XP) (Beck 2000, Beck 1999) in 1999, although according to Larman and Basili (2003) the Dynamic Systems Development Method (DSDM) published in 1997 (Stapleton 1997) was the first method that can be claimed as agile. Since the late 1990s several agile methods have surfaced, XP and Scrum (Schwaber & Beedle 2002, Schwaber 2004) being perhaps the most well-known of them. Methods such as Adaptive Software Development (ASD) (Highsmith 2000), Dynamic Systems Development Model (DSDM) (Stapleton 1997), Feature Driven Development (FDD) (Palmer & Felsing 2001) and Lean Software Development (Poppendieck & Poppendieck 2003, Poppendieck & Poppendieck 2007) have also emerged.

According to Conboy (2009), agile methods have been received well in the field of Information Systems Development (ISD) and there is strong anecdotal evidence suggesting that both the awareness and the use of these methods are highly prevalent in the field of ISD. A study conducted by Forrester Research in 2010, reported in (West et al. 2010), provides support for this claim by showing that agile development has joined the mainstream of software development methods, with 45% of organizations using a development approach that can be labelled as agile. However, the claims of agile being a mainstream development approach have been criticized. For example Stavru (2014) analysed nine different studies of the use of agile methods and concluded that only one of them could be considered as a scientific contribution, whereas the others lacked sufficient reporting and hence had low trustworthiness. The majority of the studies were from the companies providing tools for agile development and agile consulting, that could have vested interests in conducting the studies and may not follow rigorous scientific methods while conducting them. This suggestion is in line with Rodríguez et al. (2012), who similarly claimed that the majority of the studies are conducted by companies rather than academics.

[1] http://www.versionone.com/
[2] http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf

The core of agile methods is condensed into a set of four different values defined in the Manifesto for Agile Software Development (Agile Manifesto 2001). The values are described in Figure 3. Agile methods emphasise the factors on the left but, however, do not neglect the ones on the right hand side.

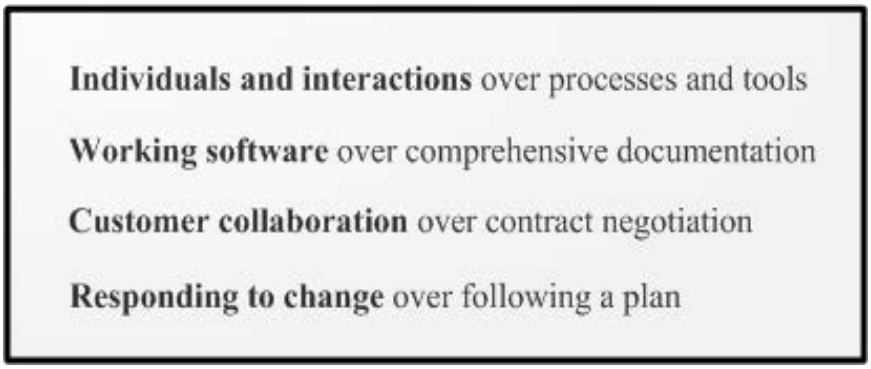

Figure 3. The agile values defined in the Agile Manifesto (2001).

In addition to the values, the Agile Manifesto (2001) defined 12 principles that should be embedded into any practices of the development approach considered as agile. Laanti et al. (2013) also discussed the areas emphasized by these principles. The principles and the areas they emphasize are described in Table 1.

Table 1. The 12 principles of agile development and what they emphasise according to (Laanti et al. 2013).

Number	***Description***	***Emphasis on***
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Continuous and early delivery, value, customer satisfaction.
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Competitiveness, adaptability, customer benefit.
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	Frequent deliveries.
4	Business people and developers must work together daily throughout the project.	Collaboration.

5	Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.	Good environment, motivated individuals, support and trust.
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Communication, efficiency
7	Working software is the primary measure of progress.	Measure progress via deliverables.
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	People, sustainability.
9	Continuous attention to technical excellence and good design enhances agility.	Technical excellence.
10	Simplicity- the art of maximizing the amount of work not done- is essential.	Optimize work, simplicity.
11	The best architectures, requirements, and designs emerge from self-organizing teams.	Self-organization.
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.	Built-in improvement of behaviour and efficiency.

Turk et al. (2005) stated that the Agile Manifesto is the condensed definition of goals and values of agile software development and that it is further detailed through these abovementioned principles. These principles on their behalf can be seen as a set of rules and policies that the processes claiming to be agile should support (Turk et al. 2005). However, the Agile Manifesto has been criticised by Conboy and Fitzgerald (2004), who perceived that it is insufficiently anchored to theoretical concepts.

Agile development has evolved from the personal experiences and collective wisdom of thought leaders and consultants from the field rather than from academia (Dingsøyr et al. 2012). After 10 years of defining the agile manifesto, the following success factors for agile projects were identified[3]:

1. **Demand technical excellence.** This is an essential factor and should be demanded by the management, the business itself and the development teams.

[3] These factors are not scientifically evaluated, but are presented in http://msdn.microsoft.com/en-us/library/hh350860(v=vs.100).aspx#intro

2. **Promote individual change and lead organizational change.** Agile methods promote response to change, which should be followed at the organization level and which should react to changing customer needs.

3. **Organize knowledge and improve education**. There is a large body of knowledge related to teams and productivity that might remain unknown to most managers and many developers. This information is encapsulated in agile practices, which should be supported by management and taught in universities educating software professionals.

4. **Maximize value creation across the entire process**. Everyone in the organization needs to be trained and educated on how to optimize performance throughout the value stream. This enables the values and principles stated in the Agile Manifesto to become fully realized.

According to Laanti et al. (2013), these targets lack much of the original emphasis defined in the twelve agile principles. What is left without mention are customer satisfaction or benefit, motivated individuals, competitiveness, environment, support and trust, collaboration and communication, emphasis on people, simplicity, sustainability and self-organization.

2.3.1 The origins of agile methods

According to a systematic review conducted by Dybå and Dingsøyr in 2008 (Dybå & Dingsøyr 2008), the majority of the included studies approached agile development as completely new concept. However, Merisalo-Rantanen et al. (2005, p. 55) labelled XP in their study as a "new bag of old tricks", indicating that some of the key ideas behind agile methods are significantly older. Larman and Basili (2003) reported the recollections of Gerald M. Weinberg considering the use of incremental development already in 1957. Further, Larman and Basili (2003) stated that many examples of Iterative and Incremental Development (IID) come from the 1970s and 1980s, which they saw as the most active and yet the least known era in the history of IID. However, the iterative approach appears not to be the only agile element introduced or proposed before the emergence of agile methods themselves. Larman and Basili (2003) discussed the NASA´s early 1960s project Mercury, in which not only an iterative approach was taken, but also the practice of "Test-first", in which tests were planned and written before each iteration, was applied. This practice is used in Extreme Programming (Beck 2000), which emerged almost four decades later.

Further, Gladden (1982) provided three propositions for software development that can be seen as having similarities to agile approaches. The first, "system objectives are more important than system requirements" translates according to Abbas et al. (2008) to the agile idea of having an overall understanding of the system instead of having a detailed set of requirements which will change during

the project. Perhaps contradictorily, Gladden (1982) stated that the main reason behind any "software fiasco" lies within a non-existent, incomplete, vague or poorly thought out set of requirements. This first proposition is also related to the shorter planning horizon promoted in agile methods, e.g. (Beck 2000, Schwaber & Beedle 2002). As Gladden (1982) claimed, objectives of the system could be set for a shorter period of time and when they are set, they are less subject to change. Further, if objectives will change this means that the system to be implemented is no longer the same as anticipated and the need for a new system needs to be examined. Concentrating on system objectives instead of requirements can, according to Gladden (1982, p. 37), "go a long way to prevent a system from "evolving" into one that the user does not want or need".

The second proposition, "a physical object conveys more information than a written specification", translates according to Abbas et al. (2008) directly to the agile value of *Working software over comprehensive documentation*. Gladden (1982) promoted the use of mock-ups early in the project and the rationale of this second proposition can be condensed in Gladden's (1982) statement that the system itself is the best means to convey the meaning of a system and the best mechanism for solidifying its concepts. Similar suggestions considering the use of prototypes were given e.g. by McCracken and Jackson in (1982). They suggested that a prototype should be built early in the development process and that this prototype should respond to the early feedback of the user. A series of prototypes or modifications to the first prototype will lead to the final product itself in a gradual fashion. (McCracken & Jackson 1982)

The third proposition, "system objectives and physical demonstrations will result in a successful product" amalgamates the other propositions and aims towards a product that satisfies the customer's needs and performs the expected functionalities. This last proposition aims towards providing the customer what is expected by focusing on system objectives and demonstrations. A similar customer-oriented approach with a focus on short term planning and early demonstrations is also present in the agile methods, e.g. (Beck 2000, Schwaber & Beedle 2002).

In 1982 McCracken and Jackson (1982) discussed the need to involve the end-customer in all phases of software development processes, not just in requirements specification alone. McCracken and Jackson (1982, p.32) suggested that "development proceeds step-by-step with the user, as insight into the user's own environment and needs is accumulated". This viewpoint is similar to the agile proposition of active customer involvement throughout the development project (Beck 2000). In addition, they suggested that the "inevitable" nature of changes should be taken into account in system development methods due to the fact that the "user, and his or her needs and environment, change during the process" (McCracken & Jackson 1982, p.31). Agile methods on their behalf recognize the possibility of changes during the project and even embrace them (Beck 2000).

2.3.2 The nature of agility

Despite the defined values and principles of agile software development, existing literature shows that the nature of what *agility* itself is seems to be vague. For example, van Oosterhout et al. (2006) stated that despite the work conducted in the field of agility, there is still a lack of consensus on its definition. In a similar vein, Conboy (2009) stated that the concept of agility in the field of Information Systems Development (ISD) is confused. For example, Conboy (2009, p. 330) claimed that "agility as a concept is highly multifaceted and has been used by many different people to refer to very different phenomena." Conboy (2009, p.330) sees this disparate nature of the different agile methods as challenging and illustrates his point by describing XP as a method providing "prescriptive operational instructions for developers", whereas Scrum resembles more project management methods than an ISD method. Lean Software Development (Poppendieck & Poppendieck 2003, Poppendieck & Poppendieck 2007) for its part can be best described as a set of philosophical guidelines (Conboy 2009). In fact, Conboy (2009) suggested that the lack of conceptual foundation regarding agility presents significant problems for practice in the field of ISD. For example, different agile methods provide completely opposite advice, which makes it difficult and confusing for teams to achieve agility (Conboy 2009).

In fact, the confusion related to agile has been noted previously. In 2005, Börjesson and Mathiassen (2005) pointed out the need for elaboration and further development of agile methods from both software developers and researchers in their study related to Software Process Improvement. Furthermore, Conboy and Fitzgerald (2007) pointed out the lack of cohesion and rigour in the usage of the concepts of agility. For example, they reported that there are multiple definitions and that researchers often use different terms to refer to the same concepts or use the same term to refer to different concepts. This same problem was discussed later by Conboy in (2009). Qumer and Henderson-Sellers (2008) also shared the perception of the vague nature of agile development. They stated that there is little evidence indicating to what extent different "so-called agile methods" meet the agility criteria envisioned in the Agile Manifesto (2001). Sarker and Sarker (2009) agreed with Conboy (2009) regarding the multifaceted nature of agile and further proposed that agility should be seen from the perspectives of resource agility, process agility and linkage agility in the context of distributed development. In this work, resource agility has its basis in the teams' access to relevant technological and human resources. Process agility for its part relates to the agility originating in the systems development method that guides the teams, including its work practices that enable collaboration across time zones. Finally, linkage agility refers to the nature of relationships within the distributed development and with stakeholders relevant to the project. This aspect also includes cultural and communicative elements. (Sarker & Sarker 2009) Abrahamsson et al. (2009) summarized the vague nature of agile concepts by arguing that each organization using agile methods needs to define what agility means to them in their context. In a similar

vein, Mahanti (2006) stated that there is no universally applicable agile method and they must be tailored to integrate with the existing processes.

Indeed, there appear to be several definitions of agility. For example, Kettunen (2009) explored the concept of agility and identified 17 different definitions that had emerged in the course of 13 years (1995–2008). In addition, Kettunen (2009) discussed several areas of agility indicating that agility is not a concept limited to software development alone. Kettunen (2009) identified the following areas of agility in his work: Strategic agility, business agility, enterprise agility, agile organization, agile workforce, IT agility, agile manufacturing and agile supply chains. In the following, some of the definitions of agility in the context of software development are discussed in order to illustrate the multifaceted nature of this concept.

For example, Erickson et al. (2005, p. 89) provided a definition for agility by stating: "at its core, agility means to strip away as much of the heaviness, commonly associated with traditional software-development methods, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines, and the like". Some authors have approached agility from a philosophical perspective. For example, Williams and Cockburn (2003) condensed agility as being "about feedback and change", whereas Highsmith and Cockburn (2001) emphasized the "soft" aspects by claiming that the central idea behind agility is to recognise that people are the primary drivers of project success together with an intense focus on manoeuvrability and effectiveness. Other definitions emphasize that the core of agility is a set of light but sufficient rules for the project, people and communication as Cockburn did in (2002). In their study, Qumer and Henderson-Sellers (2006, p. 505) defined agility as "*a persistent behaviour or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment*." Conboy for his part arrived at a definition of agility in iterative fashion and by adjusting his definition based on the existing literature in (2009). His final definition of agility was as follows: *"the continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment."* (Conboy 2009, p. 340).

Considering the plethora of different definitions of agile software development and agility, Laanti et al. (2013) discussed in their work focusing on the subject that what is understood as "agile" has evolved over time, and that due to the abstract nature of the concept "agile software development" it will probably take more time before it is fully understood. Their final conclusion on the matter was that different people mean different things when discussing agile software development and agility in general. (Laanti et al. 2013)

2.3.3 Commonalities and differences between traditional and agile methods

Although there are varying opinions concerning what actually constitutes agility, several authors have identified characteristics that are common to all agile methods. Table 2 presents the common characteristics of agile methods based on the literature.

Table 2. The characteristics of agile methods.

Characteristic	*Description*	*References*
Incremental	Software is implemented in small increments.	(Boehm & Turner 2003b, Abrahamsson et al. 2002, Abbas et al. 2008)
Iterative	Software is implemented in short development iterations.	(Boehm & Turner 2003b, Abrahamsson et al. 2002, Abbas et al. 2008)
Co-operative	All the involved parties work closely together and communicate actively.	(Boehm & Turner 2003b, Abrahamsson et al. 2002)
Straightforward	The method itself is easy to follow and to modify. The method is well documented. Principles, work structures and processes are reorganized during the project rather than being predetermined at an early stage.	(Boehm & Turner 2003b, Abrahamsson et al. 2002)
Adaptive	Changes can be made even at the last moment.	(Abrahamsson et al. 2002, Abbas et al. 2008)
Self-organizing	Teams determine the best way to conduct the required work.	(Boehm & Turner 2003b)
People oriented	People are the primary drivers of success.	(Abbas et al. 2008, Highsmith & Cockburn 2001)

As suggested by these attributes, agile methods are based on the fundamental assumption that adaptive and high quality software can be implemented incrementally by small development teams driven by the principles of continuous design improvement and testing based on rapid feedback and change. (Nerur et al. 2005). On the other hand, traditional methods approach software development efforts through more rigid characteristics. According to Nerur et al. (2005), the fundamental assumptions of software development when following traditional methods are that systems are fully specifiable and predictable, and that they can be constructed by meticulous and extensive planning. The customers' role in tradi-

tional methods is important and customer interaction is conducted on an as needed basis using formal communication focusing on contractual matters. (Boehm & Turner 2003b, Nerur et al. 2005) By contrast, in agile methods the customer's role is critical and customer interaction is driven by focused increments and communication is informal in agile development (Boehm & Turner 2003b, Nerur et al. 2005).

Despite the abovementioned differences, there are commonalities between traditional and agile methods in how they implement certain software engineering principles. For example Jian and Eberlein (2009) identified principles which they labelled as fundamental in software engineering based on the existing literature and analysed, again based on the literature, how both traditional and agile methods fulfil these principles. The analysis included from the traditional side the Waterfall, the Spiral model and the Cleanroom approach (Prowell et al. 1999). The agile methods included were XP, Scrum, DSDM and FDD. Table 3 presents a summary of the analysis conducted in (Jiang & Eberlein 2009).

Table 3. How various software engineering principles are implemented in traditional and agile methods (Jiang & Eberlein 2009).

Software engineering principle	***Examples of traditional software engineering practices***	***Examples of agile software development practices***
Invest in problem understanding	Identify requirements and functions (Cleanroom) Requirements specification development (Waterfall) Determination of requirements (Spiral model)	Product backlog (Scrum) Sprint backlog (Scrum) User story card (XP)
Inspect code	Peer review of individual work (Cleanroom) Code should be reviewed by a party not involved in writing the original code (Waterfall)	Pair programming (XP) Inspections (FDD)
Involve the customers	Maintain customer involvement in specification and certification (Waterfall) Designers must communicate with interface designers, with management and the customers (Waterfall)	Active user involvement and stakeholder collaboration (DSDM) On-site customer (XP)

Risk management Identify and manage uncertainty	Determination of plans (Spiral model) Identification and resolution of risks during the entire software development process (Spiral model) Rapid revision of incremental plans for new requirements and response to schedule and budget changes (Cleanroom)	Regular build schedule (FDD) Planning game (XP)
Iterative and incremental development	Incremental development cycle (Cleanroom) Iterative and incremental development (Spiral model)	Iterative and incremental development. Scrum, DSDM and XP mentioned explicitly in (Jiang & Eberlein 2009). According to (Boehm & Turner 2003b, Abrahamsson et al. 2002, Abbas et al. 2008) all agile methods are iterative and incremental.
Use better and fewer people People are the key to success	Testing and development are carried out by small teams (Cleanroom)	Empowering teams (DSDM)
Establish a software process that provides flexibility	Team frequently reviews and discusses design strategies (Cleanroom)	Daily scrums (Scrum) Open workspace (XP) Reporting/visibility of results (FDD)
Change is inherent to software; plan for it and manage it	Requirements management (Spiral model)	Baselined requirements (DSDM)
Maintain disciplined product control	Configuration management (Spiral model)	Configuration management (FDD)
Manage quality throughout the life-cycle as formally as possible	Team strives for design simplification (Cleanroom)	Simple design (XP)
Produce software in a stepwise fashion.	Stepwise system integration process (Cleanroom)	Continuous integration (XP) Integrated testing (DSDM)

Based on the above, agile and traditional methods both fulfil the same development principles but approach development from different angles. Whereas traditional methods approach project development from a strict and rigid perspective, agile methods provide development organization more freedom to achieve the intended target of the project, namely a product that meets the customers' needs. Furthermore, the comparison presented in Table 3 suggests that the ideas behind agile methods are hardly unique.

Traditional and agile methods are not mutually exclusive. Instead, they can be combined in order to achieve the best possible development approach. Boehm & Turner (2003b) proposed using risk analysis as a tool for balancing agility with the discipline provided by traditional approaches. This can help to identify the most suitable approach for the project, which in turn increases the probability of success. Boehm and Turner (2003b) promoted the idea of combining agile and traditional approaches based on their claim that both agility and discipline are required for sustainable, successful software development. In a similar vein, Turk et al. (2005) suggested that in order to use agile processes appropriately it is necessary to have an understanding of the situations in which agile processes are, or are not, applicable. Ramesh et al. (2006) also discussed balancing agility with more discipline regarding the five key challenges which they identified. These challenges were as follows: *1) Communication need vs. communication impedance, 2) Fixed vs. evolving quality requirements, 3) People- vs. process-oriented control, 4) Formal vs. informal agreement* and *5) Lack of team cohesion*. The last of these challenges is related to improving team cohesion in globally distributed development, but considering the first four issues it is suggested that the balance between the elements (e.g. fixed vs. evolving quality requirements) should be sought in globally distributed agile development.

The needs that drive both small and large organizations to adopt agility have been discussed e.g. by Lindvall et al. (2004). According to these authors, traditional software development methods are seen as too bureaucratic, cumbersome and inflexible, in addition to the pressure to produce more at lower cost. These are some of the factors that have led companies to seek alternative ways of working in form of agile methods. Furthermore, Lindvall et al. (2004) summarized that problems related to inefficient management of projects with unspecified requirements, to gaining deeper understanding of end users' real needs, and the problem of specifications being outdated by the time they are finalized are also seen among the driving forces towards agility. Furthermore, rapid changes in the requirements and in the market and in technology necessitate a more flexible process that is capable of adapting to changing requirements (Lindvall et al. 2004).

2.3.4 Benefits and challenges of agile methods

Since the inception of agile methods, a significant amount of work has been conducted about their stated benefits. For example Dybå and Dingsøyr (2008) cited studies which reported benefits from the areas of handling defects, learning in pair

programming, estimation, customer collaboration, improved focus on current work for the engineers and thinking ahead for managers. Laanti et al. (2011) studied the benefits of agile methods at Nokia Oy and concluded that most of their 1000 respondents experienced higher work satisfaction, increased feeling of effectiveness, increases in quality, transparency, happiness and autonomy, and earlier defect detention. Petersen and Wohlin (2009) also studied the advantages of agile methods in their work conducted at Ericsson AB. They identified benefits and mapped them against the already known positive findings that were discussed in the State-of-the-Art section of their work. They concluded that the more precise and easily estimated requirements stemming from small scope encountered in their study were in line with previous findings. In addition, early feedback resulting from frequent deliveries, reduced need for documentation due to increased direct communication, reductions in the amount of rework and increased transparency of responsibilities aiming towards higher quality were also experienced as benefits supporting previous work. (Petersen & Wohlin 2009) They also found additional benefits. The volatility of requirements was low due to prioritization and their limited scope, and the work that was started was always completed. (Petersen & Wohlin 2009) In 2008, Salo and Abrahamsson (2008) conducted a survey within 35 individual projects from 13 organizations located in eight countries. They focused on studying the use and usefulness of XP and Scrum in the embedded software development organizations. The results indicated that 54% of the respondents mentioned that they applied XP practices either systematically, most of the time or at least sometimes. The most commonly applied XP practices were (1) open office space, (2) coding standards, (3) 40 h week, (4) continuous integration and (5) collective code ownership. The on-site customer practice was applied either systematically or most of the time in 24% of the studied projects, whereas the use of this practice was either “rarely” or “never” in 42% of the projects.

Considering the practices, nearly 90% of those who applied XP practices saw them in a positive light (extremely useful, very useful or useful) and the most appreciated practices were Collective Code Ownership, 40 hour week, Coding Standards and Simple Design. Negative findings (not useful or harmful) were also identified within 5.8% of responses and the least appreciated practices among those who applied them were Pair Programming (20% negative responses), Test Driven Development (12%) and On-Site Customer with 11% negative responses. (Salo & Abrahamsson 2008)

Considering Scrum, applying the practices of Scrum was either systematically, mostly or sometimes used by 27% of the respondents, Product Backlog being the most favoured practice with 24% of the respondents applying it either systematically or most of the time. Despite the lower adoption rate of Scrum practices compared to XP, 77% of the responses of Scrum’s usefulness were positive whereas 11% were related to negative experiences. (Salo & Abrahamsson 2008) More recent findings from the industry presented in VersionOne’s 7th annual survey on the state of agile from 2013, referenced earlier in this work, indicates that Scrum is the most widely adopted agile method in the industry, with a 54% share. This survey indicated that the top five agile practices applied were Daily Standups,

Iteration Planning, Unit Testing, Retrospectives and Release Planning. In addition, the top five benefits gained from using agile methods were:

1. Increased ability to manage changing priorities.
2. Increased productivity.
3. Improved project visibility.
4. Improved team morale.
5. Enhanced software quality.

Agile methods also pose challenges. For example Mahanti (2006) discussed the issues encountered at personal levels. These issues include resistance to and fear of change, and closed-minded attitudes causing people automatically to reject and even deliberately to spread false information about agile methods. In addition, the misunderstanding that agile and traditional approaches cannot be combined, and that if an agile method is adopted it must be adopted at once in its entirety, can be problematical. A long history of working with documentation- heavy sequential approaches was identified as a challenge, as well as outdated skills and knowledge in the context of new development approaches. Since agile methods promote self-organization and cross-functionality of the team members, highly specialized skills in narrow areas of expertise can create challenges in the agile environment. (Mahanti 2006) One of the challenges specifically identified in upper management is reluctance to surrender the feeling of control resulting from traditional development artefacts, such as Gantt-charts. In addition, upper management may be troubled if product commitments can be made in agile environment. One central challenge is also that upper management fears that agile projects will go on forever. Tracking progress in the agile environment may cause doubts in upper management, along with the impact of agile adoption into other groups of an organization. (Cohn & Ford 2003)

A major challenge seems to lie in changing the whole organization, as opposed to the particular development team in question (Nerur et al. 2005, Grossman et al. 2004). According to Grossman et al. (2004) the challenge is to adapt and reconcile both the agile and corporate culture processes and methods without compromising either of them. This process is easier if there is a significant buy-in from all the stakeholders. Nerur et al. (2005) specified that organizations should carefully assess their readiness for agile methods and continued that organizational forms and cultures supporting innovation are friendlier environments for adopting agile methods than organizations built around formalization and bureaucracy. This is an important viewpoint considering that bureaucratic organizations have been identified as difficult environments for agile projects to succeed (Berger 2007).

From the industry's side the five most significant challenges[4] in further adopting agile methods were related to the ability to change organizational culture (52% of the respondents), general resistance towards change (41%), trying to integrate agile elements into non-agile framework (35%), availability of personnel with suitable skills and both customer collaboration and project complexity (26%).

Originally, agile methods were developed in the context of small collocated independent teams with constant access to customers. Applying agile development in large scale development has been found to be problematic. Boehm (2002) claimed that agile methods work well if future architectural concerns do not need to be taken into account, requirements are not defined upfront and are highly subject to change, committed customers are readily available, project organization is kept small and involves skilful personnel, and the domain is non-safety critical. Otherwise, in more complex environments and larger projects Boehm suggested applying more plan-driven approaches. (Boehm 2002)

Lindvall et al. (2004) stated that within large organizations projects need to interact with the rest of the organization and follow its overall rules, standard processes and quality systems. Furthermore, the software being developed can be a part of a larger system developed together with other teams, and the work conducted by the teams should integrate smoothly. In order to overcome these hurdles, organizations need to better define the interfaces between the organization and the agile teams. Along similar lines, Kettunen and Laanti (2008) concluded that in order for companies to benefit from agile development, a more holistic approach needs to be taken. For example, reactiveness to changes just when they arise is not enough since the impacts of these changes need to be considered in the wider contexts, such as e.g. product lines. (Kettunen & Laanti 2008)

Tailoring agile methods to meet the needs of the environment can be seen as a strategy for getting the most out of them. Further, according to (Rizwan Jameel Qureshi 2012) there appears to be a contradiction between traditional and agile development approaches. Traditional methods appear not to fulfil the software industry's need for rapid development without compromising quality. On the other hand, agile methods cannot be delivered in medium-size and large development efforts as such, due to inadequate documentation, lack of risk management and lack of focus in the architectural aspects. (Rizwan Jameel Qureshi 2012)

In 2002, Bowers et al. (2002) described how XP was tailored for the needs of an effort making changes to large mission critical software. The results were examined via selected XP practices and provided some evidence that agile methods can be applied in life-critical system development. Cao et al. (2004) presented agile practices which they considered suitable in complex large scale development. Kähkönen (2004) approached scaling agile from the perspective of development teams. In this study, Kähkönen argued that agile development methods do not take into account issues spanning over several development teams and proposed regular workshops involving people from different parts of the organization

[4] http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf

resolving these issues. Large organizations using agile methods should hence have overlapping communities of teams. (Kähkönen 2004)

Whereas these studies focused on tailoring individual practices and elements of agile methods, Fitzgerald et al. (2006) described how XP and Scrum were merged to support development at Intel Shannon in Ireland. The study indicated that by moving away from the textbook versions of named approaches through careful evaluation of the elements of agile methods, adjusting them based on confirmed needs and even discarding those seen as unnecessary can create positive outcomes. Similarly, (Rizwan Jameel Qureshi 2012) presented an extended version of the XP method with distinct phases for project planning, analysis and risk management, design and development and testing in order for XP to overcome the deficiencies of the method in large and medium-sized projects.

2.3.5 Critique of agile methods

Agile methods have also been criticized. For example, Turk et al. (2005) found limitations in agile methods that they inferred from assumptions underlying the principles[5] of the methods (Agile Manifesto 2001). In their work, Turk et al. (2005, p.7) defined assumptions as "premises or beliefs that are taken for granted and are not expected to be proven".

According to Turk et al. (2005) agile methods have *limited support for distributed development environments* due to assumptions related to close customer interaction, collocated development teams, the ability to communicate face-to-face and the counter-productive nature of creating extensive documentation and models of the developed software. These limitations stem from geographical and temporal distances that can be significant in distributed environments. These distances and the challenges they pose, along with identified solutions for mitigating them, are discussed later in this work. Agile methods also have *limited support for subcontracting*, since subcontracting often requires precise contracts stipulating what is expected from a subcontractor (Turk et al. 2005). Because of this, the assumptions related to close customer interaction, collocated teams, face-to-face communication, reliance on tacit knowledge and continuously evolving requirements may not hold. As a solution, Turk et al. (2005) proposed that companies either increase the amount of documentation or require subcontractors to co-locate with them.

The third limitation encountered was *limited support for development involving large teams.* Agile methods are targeted towards small to medium-sized teams, to which the "management-in-the-small" approach can be applied in terms of coordination, control and communication. Large teams can often have many sub-teams of specialists and are often dispersed across different locations. Hence, the assumptions of close customer involvement, collocated teams, face-to-face communication and reliance on tacit knowledge do not hold. Considering large teams, more traditional software engineering practices related to documentation, change

[5] http://agilemanifesto.org/principles.html

control and architecture-centric development are more appropriate. (Turk et al. 2005) Further, agile methods have *limited support for building reusable artefacts*. If the amount of documentation is minimized, determining when and where a particular artefact could be reused is challenging. Furthermore the nature of agile methods, being focused on solving a particular problem rather than creating generalizable solutions, causes limitations. Continuous redesign (refactoring) is also difficult when not creating application-specific artefacts. This decreases the opportunities for customer feedback and hence improvements in design and quality are reduced. Therefore, the assumptions related to minimized amount of necessary documentation, restricting evaluation of software artefacts to frequent informal meetings, reviews and code testing, not targeting towards generalizable solutions and continuous refactoring might not hold. Addressing this limitation requires more rigid procedures related to, for example, documentation.

Agile methods also provide *limited support for developing safety critical software*, since assumptions related to reliance on tacit knowledge, continuous refactoring and relying on informal quality assurance techniques may not be valid. Again more formal procedures should be applied and more formal specifications created. Finally, there is limited support for developing extensive, complex software in which assumptions related to tacit knowledge instead of detailed communication, informal quality assurance, developing software within short iterations, and refactoring may not be valid. (Turk et al. 2005). In order to address this limitation, Turk et al. (2005) postulated that the use of the Model Driven Architecture approach, for example, could be useful. Fitzgerald et al. (2013) also addressed the issue of scaling agile methods to safety critical systems in regulated environments, such as the automotive, aviation and financial industries that have to comply with formal standards, guidelines, directives and regulations. In their work, Fitzgerald et al. (2013) described how an agile approach was successfully implemented in such environments. In particular, they paid particular attention to the concepts of continuous compliance and living traceability. The first refers to the practice of conducting independent quality assurance audits at the end of each Sprint and the latter refers to complete transparency of the development process at any given time. (Fitzgerald et al. 2013)

A study conducted by Petersen and Wohlin (2009) aimed towards identifying both advantages and issues in agile development. They conducted a case study within Ericsson AB, which is a large multinational company focusing on telecommunication solutions. They studied three sub-system projects that were part of a large-scale agile development project. The findings were extracted from a total of 33 individual interviews. For the purpose of the study they conducted a state-of-the-art study on both advantages and issues of agile development. Table 4 summarizes their findings related to encountered issues. In addition, it is indicated whether a particular finding confirms an earlier identified issue or whether it is completely new.

Table 4. Summary of issues encountered in agile development and their descriptions based on Petersen and Wohlin (2009).

Issue	*Description*
Testing lead times and maintenance	Latest system version cycle times may extend lead-time for package deliveries if a software package is rejected by testing due to quality issues or if it is not ready. In such cases, the package has to wait for the next cycle before it can be integrated. Secondly, more frequent releases increase maintenance efforts. This occurs when there are multiple different versions of the product on the market which need to be supported and test environments for these versions have to be recreated. This issue confirms the earlier finding that realizing continuous testing requires significant effort, since creating an integrated test environment is difficult for different platforms and system dependencies.
Management overheads and coordination	Results from a large number of teams requiring extensive coordination and communication. This issue confirms the earlier finding that agile methods do not scale well and are difficult to scale.
Little focus on architecture	Dependencies that are rooted in implementation details are difficult to identify and are not covered in the plan. What makes this particularly challenging is that projects implementing different packages have no control over other packages being developed, which prevents early identification of dependencies. This finding confirms the earlier finding of architectural design not having adequate focus in agile development, which leads to bad design decisions.
Requirements prioritization and handover	Complex decision making processes involving several people that must be involved in these processes causes requirements processing to take a lot of time. Consequently, complex decision-making processes affect teams which have to wait for the requirements in order to create a product backlog. Getting the requirements priority list right is challenging since the list itself must be agile in order to reflect changing customer needs. This finding was not discussed in the literature survey by Petersen and Wohlin (2009).
Test coverage reduction	Test coverage reduces within projects due to lack of independent testing. This finding was not discussed in the literature survey by Petersen and Wohlin (2009).

Increased configuration management effort	Configuration management must coordinate a significant number of internal releases. The number of baselines is high. This finding was not discussed in the literature survey by Petersen and Wohlin (2009).

In addition to the abovementioned, issues related to a high level of documentation and product integration put focus mainly on programming the system's configuration environment. According to Petersen and Wohlin (2009) these findings are more related to the context of the case organization and, therefore, their generalization is limited. The literature which Petersen and Wohlin (2009) studied identified issues that were not encountered in their study. These issues are as follows:

- Pair programming is seen as inefficient and exhaustive.
- Pair programming is not applicable if one of the partners is much more experienced than the other. This result is derived from the perspective of students.
- Team members need to be very highly qualified in order for an agile project to succeed.
- Communication within a team works well but inter-team communication is not effective.
- Empowerment of people makes managers initially afraid and requires sufficient training for managers.
- Technical issues are raised too early from the management point of view, since implementation starts very early.
- The role of an on-site customer is very demanding, which creates stress. The role of an on-site customer is discussed in more detail later in this work.

Petersen and Wohlin (2009) postulated the inconsistence between previously found issues and the issues which they encountered. According to them one reason for this may have been that the previously conducted work was not explicitly targeted towards issue identification. Another possible explanation is that large-

scale agile projects are difficult to implement due to increased complexity in terms of number of the projects, product size and people. In addition to the abovementioned critique, a study conducted by Solinski and Petersen (2014) summarized very high professional skill level demands, in a similar vein with Petersen and Wohlin (2009), as one of the most significant limitations of agile methods. The other two main limitations were the lack of suitability for specific product domains and scalability of agile methods. Similar findings were made earlier by Turk et al. (2005), and Petersen and Wohlin (2009) also identified the issue of scalability of agile methods, as Boehm also did in (2002).

The critique of agile methods involves both aspects related to agile methods in general and specific practices. In addition to the abovementioned practices, Test Driven Development, for example, has received criticism. For example Siniaalto and Abrahamsson (2008) concluded that the stated benefits of Test Driven Development are not self-evident and automatic. They found that in some cases Test Driven Development can lead to a more complex software structure that is difficult to change. Siniaalto and Abrahamsson (2008) concluded that the question of whether Test Driven Development ultimately improves software design, as it should, remains unanswered. Another study by Siniaalto and Abrahamsson (2007) concluded that, at least in situations in which developers are inexperienced, Test Driven Development does not always produce a highly cohesive program code, as it should according to the literature. Although it is not postulated by Siniaalto and Abrahamsson (2007), high expectations set for professional skills in agile development might contribute to this finding. Jokela and Abrahamsson (2004) criticised XP from the perspective of usability, claiming that XP pays almost no attention to usability aspects of software products, apart from certain actions that can be regarded as implicit usability evaluations. By these actions Jokela and Abrahamsson (2004) referred to the verification of software increments against design requirements and usability feedback. Instead, they concluded that the responsibility regarding usability is a responsibility of the customer. However, this does not automatically lead to poor usability. In this case, good usability is more or less a coincident, depending entirely on the intuitions of the customer and the designers involved in the development process. Further, if the customer is not interested in usability issues or does not understand them, usability aspects would then be based on the interests of the design team. Usability engineering would increase efforts and result in tasks that are not requested by the customer, which could in turn result in usability engineering not taking off. On the other hand, if the customer is knowledgeable about usability aspects this could change the role of the design team dramatically. (Jokela & Abrahamsson 2004)

2.4 Distributed software development

Developing software in a distributed fashion began in the early 1990s (Herbsleb & Moitra 2001) and at an increasing pace this phenomenon is becoming almost a business necessity for companies, e.g. (Damian & Moitra 2006). Reasons for

taking this approach result from reaping the benefits from a more affordable and skilful global workforce, potential 24 hour development and matured technical infrastructure that enables collaboration across geographical distances (Ebert & De Neve 2001, Herbsleb & Moitra 2001, Komi-Sirviö & Tihinen 2005, Gorton & Motwani 1996, Battin et al. 2001). Distributed Software Development (DSD) can be a phenomenon of global scale, when it is often labelled in the literature as Globally Distributed Development (GSD). GSD is characterised by involving stakeholders from different national and organizational cultures and time zones. (Damian 2002). Implementation of tasks at different stages of the software's lifecycle may also be separated and individually implemented at geographically dispersed locations using information and communication technologies as the means of coordinating the development efforts (Sahay 2003)

Distributed development can also be something that happens in a much narrower context. According to Prikladnicki et al. (2003), distribution can range from adjacent buildings to other continents. They proposed a model that includes this aspect, presented in Figure 4.

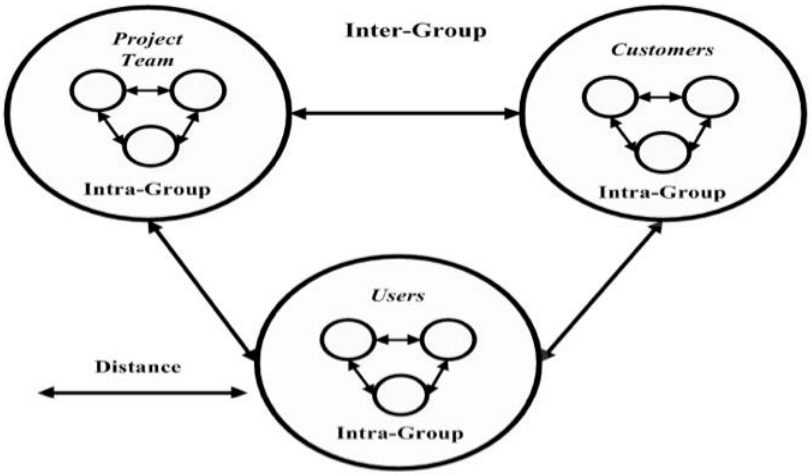

Figure 4. DSD distribution levels as presented in (Prikladnicki et al. 2003).

The model consists of groups of actors working with each other within different scenarios. The Project Team involves everyone from development, including developers, testers, business-oriented personnel, etc. Customers are the person or organization that requested the project and Users represent people responsible for providing e.g. requirements. The scenarios include same physical location in which the actors work within the same location. A Cross town scenario occurs when the actors are distributed within the same city and in the no time shift scenario they are located within the same country. A Continental scenario indicates that the actors are located on the same continent and finally, in a global scenario the actors are dispersed across continents. These distances apply to particular

actors, e.g. Users, as well as groups of actors. As an example, the distance between Customers and Users can follow a continental scenario when at the same time the distance between Project team members is global. (Prikladnicki et al. 2003). This approach therefore takes into account both DSD and GSD scenarios.

O'Conchúir et al. (2006) stated that the assumed benefits of GSD have not been extensively analysed. Based on the literature, they identified in (Conchúir et al. 2009) six claimed benefits and found that none of them were fully realized. The assumed benefits of the "follow the sun" development were not realized. Software was not developed in the follow the sun approach, but testing activities sometimes followed this approach. In addition, the claimed benefit that the diverse backgrounds of people from different sites increases innovativeness and enables sharing best practices was found to be a myth. Instead, people with lower wages were seen as a threat and information was not shared to them more than was required to get the job done. The four only partially realised benefits along with their descriptions are presented in Table 5.

Table 5. The partially realised benefits of GSD according to (Conchúir et al. 2009).

Assumed benefit	***Description of findings***
Reduced development costs	The differences in salary levels can be as much as eight-fold between different countries. These savings can be eroded by increasing complexities in coordination and in ramping up the competence levels of other sites. Control and communication overheads are significant. (O'Conchuir et al. 2006)
Cross-site modularization of work	Modularized work can be conducted independently and can reduce the need for communication between sites. If there is too much independence and lack of communication, integrating the modules later can be problematical. (O'Conchuir et al. 2006)
Access to skilled labour	Accessing a large pool of competent staff is possible. However, in some countries the rapid increase of jobs in software development has led to high attrition rates within companies. Significant issues in communication may arise. (O'Conchuir et al. 2006)
Closer proximity to customer and market	Locating development efforts closer to the target market provides better access to customers and the local market conditions. However, this approach introduces cultural challenges that need to be solved. (O'Conchuir et al. 2006)

As can be seen from Table 5, these partially realised benefits can also create challenges. This suggests that the assumed benefits are either myths or can potentially create such challenges that outsourcing is terminated or not even considered. As an example, Sutherland et al. (2009) reported that the cost savings are not always as expected and as a result, outsourcing efforts can be permanently terminated. The high attrition rates up to 30–50% annually can also lead to decisions of not outsourcing work to distributed partners. (Sutherland et al. 2007)

According to Šmite et al. (2010c) the field of GSD is still immature, focusing more on challenges of GSD instead of, in particular, empirically evaluated solutions, and there is no standard recipe for successful GSD efforts. In addition, in the systematic review which they conducted (Šmite et al. 2010c), there are only a few clear stories of both success and failure from which one can learn. Šmite et al. contemplated the reasons behind this finding and suggested that GSD is itself so complex that identifying the exact reasons behind failure and success is very difficult. Further, the definition of success or failure was dependent on what was analysed in the works included in the systematic review. For example, a success story refers to an overall success of the project whereas a successful practice can be a reference to something that was successful despite the overall failure of the project. Šmite et al. (2010c) also applied this same reasoning when failures were discussed and concluded that the lack of clear reasons behind especially failures may be ambiguous due to the company image; sharing stories of failure may affect how they are perceived by the public.

Whereas Šmite et al. (2010c) discussed the elusive nature of reasons behind success and failure, the work focusing on challenges in distributed development has identified many of them. For example, the systematic review conducted by Jiménez et al. (2009) identified these hindrances:

1. **Communication.** A large amount of communication is required between the stakeholders involved in the development effort. Much information is transferred through various tools in various formats and without following communication standards, hence leading to misunderstandings and increased response times. These drawbacks in conjunction with large and changing personnel networks and complex infrastructure negatively affect communication frequency and quality, hence decreasing productivity.

2. **Group awareness.** Feelings of indifference and isolation have a negative effect on the productivity of the members of distributed teams.

3. **Software configuration management.** Source code control can be problematic in distributed environments. Issues with synchronization and coordination become increasingly complex as the degree of distribution within the team grows.

4. **Knowledge management.** Knowledge management can be challenging in distributed environments. Hence, effective information sharing systems

need to be in place in order for the experiences, decisions and skills of the team members and their predecessors to be effectively accumulated.

5. **Coordination.** In distributed development coordination is more challenging due to problems resulting from communication, lack of awareness and the complexity of the organization itself. These challenges affect how the work needs to be managed and structured.

6. **Collaboration.** By its nature, software development is a collaborative activity requiring cooperation between various different stakeholders. Distributed development can make this collaboration difficult and hence tools supporting work over geographically distributed sites are required.

7. **Project and process management.** Distributed environments make task assignments, scheduling and cost estimation more challenging. This results from volatile requirements, changing specifications, lack of informal communication and cultural diversity.

8. **Process support.** Distributed environments often involve a wide network of autonomous, heterogeneous and distributed models. These should support coordination and cooperation of the teams and offer an automated support to distributed project management. Problems related to tools also appear within this context.

9. **Quality and measurement.** The quality of the processes which distributed projects apply has a significant impact on the product quality. In distributed efforts, the impacts of issues encountered can be magnified and it is more difficult to recover from the problems than in the case of collocated projects.

10. **Risk management.** Distribution creates additional challenges in risk management in the form of coordination issues, problem resolution, knowledge sharing, risk identification and evolving requirements.

In addition, da Silva et al. (2010) conducted a systematic review of the challenges, best practices, models and tools in DSD project management. They found altogether 30 challenge categories and according to the number of studies reporting challenges in identified categories, the TOP 5 challenges affecting DSD project management appear to be the following:

1. **Effective communication** with 34 references.

2. **Cultural differences** with 31 references.

3. **Coordination** with 23 references.

4. **Time zone differences** with 13 references.

5. **Trust** with 13 references.

Thirteen occurrences were also found related to **asymmetry in processes, policies and standards** as well as to **physical distance**.

Based on the findings considering the challenges, communication-related issues appear to be the most common obstacles hindering distributed development. Perhaps as an attempt to answer the many challenges of distributed development, companies are reconsidering the ways of distributing development work. In this context, the terms “farshoring” and “nearshoring” should be discussed.

The phenomenon behind engaging globally distributed development is often labelled “offshoring”. According to Niederman et al. (2006), offshoring is about distributing the work to people outside the borders of the host country. Jiménez et al. (2009) added that offshoring also includes the cost factor, namely cheaper human resources. The term is also associated with distance and is often connected with offshoring the development work to India, in particular. Due to significant distance between the host countries, such as the US (US-India relationships being the most reported according to (Jalali & Wohlin 2012), this relationship is also referred to as “farshoring”. This distance is divided into three main categories by Noll et al. (2010), who stated that geographical, temporal and cultural distances are the main factors hindering global software development. In addition to the abovementioned, Noll et al. (2010) identified on the basis of existing literature that language barriers, issues with trust and fear of losing jobs, and problems with managing the distributed efforts, just to mention a few, all play their role in GDS. Due to the challenges encountered in offshoring, “nearshoring” has been of interest more recently. This term refers to an approach in which the jobs are transferred to countries closer within geographical, temporal and cultural distances. (Jiménez et al. 2009) In this relationship, lower wages are still a driving force and the customer expects to gain benefits from one or more of the factors related to proximity (i.e. distance). These factors are temporal, cultural, linguistic, political, historical, economic and geographic proximities.

2.4.1 Distributed agile software development

Agile development methods have also been adopted to distributed settings. According to (Shrivastava & Date 2010), agile development and distributed development are both strong trends resulting from increased demands for faster and cheaper implementation of high quality products. Therefore, applying agile methods is a necessity for many organizations. (Shrivastava & Date 2010). According to Holmström et al. (2006b), it is a common view that agile methods are not applicable in globally distributed development environments. Agile methods were designed to work in collocated contexts, and following the tenets of agile development can be difficult in distributed environments. For example, the strong empha-

sis on informal, preferably face-to-face communication can be very difficult to achieve while working in a distributed fashion. (Ramesh et al. 2006).

Very soon after the inception of agile methods, studies considering their applicability in distributed environments began to emerge. In 2001, Kircher et al. (2001) concluded in their work focusing on XP that some practices of agile methods can be applied as such in distributed development, without a co-located team. Out of 12 XP practices, planning game, pair-programming, continuous integration and on-site customer were dependent on stakeholder collocation, which they saw as the key factor hindering the application of XP to distributed development. Small releases, metaphor, simple design, testing, refactoring, collective code ownership, 40-hour week, and coding standards were labelled as suitable for distributed environments. Kircher et al. (2001) proposed Distributed XP (DXP), which promotes the use of interactive communication tools to enable the use of collocation-dependent practices. In a similar vein, Maurer 2002) discussed an interactive tool MILOS that supports collaboration, coordination and communication of distributed teams using XP. Braithwaite and Joyce (2005) also discussed DXP and defined a set of practices enabling distribution in the thematic areas they identified. These areas are *people*, *communication* and *code*. The practices for the *people* theme involve maintaining a "single team identity" across all the sites involved, making all team members empowered to make decisions and equal in skills and numbers, using local representatives at remote sites to interpret the communications, demonstrating the idea of the "single team identity" and transferring domain knowledge between sites. Team members should also be rotated between sites in order to build trust between distributed partners. Under the *communication* theme, remote stand-ups and multiple communication tools should be used in order to enable DXP. Using a wiki as a shared location for asynchronous communication is also promoted. The *code* theme involves using a single shared code base to work with and in order to avoid integration issues, acceptance and/or functional tests that state the intention to work in a particular area should be published to other participants. This makes the intended code changes visible to others. More recent approaches for applying agile methods in distributed settings have been presented e.g. by del Nuevo et al. (2011). They proposed a method called scRumUP, which is a hybrid method for distributed development combining elements from both Scrum and Rational Unified Process (RUP) (Kruchten 2004).

The application of Scrum in distributed settings has also been under investigation. Although both are labelled as agile methods, XP focuses on implementation practices and Scrum on project management level aspects. Scrum is therefore an agile project development framework that does not take any programming level practices into account. Similarly to XP, challenges in applying Scrum in agile GSD have been identified. Hossain et al. (2009) conducted a systematic literature review of the field and identified seven different categories of the findings. Problems with the lack of synchronous communication were considered as one of the most important challenges encountered by projects following Scrum in globally distributed environments. In fact, it appears that most of the challenges identified by Hossain et al. (2009) are related to communication. The analysed studies indicat-

ed issues with poor technical infrastructure. For example, the findings of Paasivaara et al. (2008b) suggest that the lack of an appropriate technical infrastructure can prevent the use of videoconferencing between remote sites. Issues related to collaboration tools for remote partners were also widely reported. Hossain et al. (2009) concluded that despite its limitations, Scrum can be applied in distributed projects with multiple teams and extensive personnel. Scrum can also be applied when there are no overlapping work times between remote sites. Based on the available evidence Hossain et al. (2009) also concluded that Scrum practices need to be modified or extended in order to provide support for teams operating in globally distributed contexts. These practices and policies to be extended and modified revolve around collaboration between personnel. Additional local meetings, strict communication policies, attendance of key personnel in all distributed meetings and reducing the numbers of meetings are approaches that could be adopted in globally distributed agile development. In addition, multiple communication channels should be used. (Hossain et al. 2009) However, there are challenges which emerge from cultural differences and from both geographical and temporal distances. These can have negative impacts on collaboration, coordination and communication, that were identified as the key challenge areas of the study (Hossain et al. 2009). Further, there appear to be certain contextual factors that may limit the use of Scrum practices, for example the use of Scrum in safety critical systems development is limited. (Hossain et al. 2009) The lack of rigorous planning and formal evaluation techniques has limited the use of agile methods in the context of safety critical systems (Wolff 2012). In order to address this problem, e.g. Wolff (2012) presented a theoretical approach derived from industrial experience of applying Scrum in large mission critical systems in the domain of military aircraft self-defence systems. In this approach, the tasks implemented within iterations include both conventional tasks with no or very little uncertainty that can be implemented immediately and tasks related to investigation of formal specification tasks. These tasks are focused on modelling and validation of high risk system properties before they are implemented in the forthcoming iterations. The results of the iterations are hence twofold: both working software and working models to describe high risk properties are produced. (Wolff 2012)

As Ågerfalk et al. (2009) suggested, many of the difficulties, such as increased complexity in communication, cooperation, coordination, control and culture, as well as the tools and technology, are the same when discussing challenges in global software development and agile methods.

On the other hand, it has been found that there are certain elements in agile development that can help with challenges related to distributed environments. For example, Paasivaara et al. (2008b) identified the following benefits which Scrum provides for globally distributed efforts.

1. **Improved communication**. More structured, more open communication.

2. **Increased trust**. Remote sites are able to conduct demanding tasks. Monitoring progress through more frequent communication.

3. **Improved motivation**. Swift clarifications of questions, access to relevant people at the right time, common goals and values.

4. **Better quality**.

It is also recommended that adopting agile development in distributed settings should be conducted gradually. For example, Sureshchandra and Shirinivasavadhani (2008) proposed that the transition process should begin with a more rigid approach in which the core site driving the development has control over the project. As the project progresses and experience and comfort levels increase, remote sites are given more responsibility over their work and the work process itself becomes less formal and increasingly collaborative, involving e.g. joint meetings and overlapping work hours. Similarly to the ideas of balancing agility and discipline presented in (Boehm & Turner 2003b, Ramesh et al. 2006), Lee et al. (2006) suggested that in globally distributed projects, agile methods must be adjusted and modified so that more rigour and discipline should be introduced to the process, since without these software development can become inefficient and chaotic in global contexts. However, Estler et al. (2012) conducted a case study involving 66 distributed industrial projects which they classified either as either plan-driven or agile. Perhaps counterintuitively to the suggestions made about balancing agility and discipline, their findings indicated that both agile and plan-driven approaches can be equally effective, or ineffective, in globally distributed contexts. Therefore, the choice of method does not matter.

Šmite et al. (2010b) stated that combining agile methods and distributed development is of immense interest in the industry. Despite this popularity, understanding of the viability and the limitations of agile methods in distributed software projects is still limited, and how agile methods will play out in this seemingly incompatible environment remains an open question. Similarly, the results of the study made by Stankovic et al. (2013) suggest that defining the success factors of distributed agile development projects is an elusive target.

Šmite et al. however gave some advice in (Šmite et al. 2010a). The components of this advice are as follows:

1. Meet face-to-face, co-locate for a while, exchange team members and avoid distribution if possible.

2. Patience and stamina are needed.

3. Provide communication infrastructure and tools that support rich and intense interpersonal communication.

4. Pay attention to how people are supported and give careful consideration to distribution.

5. Make people enthusiastic and don't follow the method religiously. Instead, adopt and adjust as is justified and explain why.

6. Enable teams to deliver overall functionality by establishing location-independent teams.

7. Seek true customer involvement, overcome unavailability of the customer and maintain continuous connectivity with the customer.

8. Ensure the team members' availability to collaborate with each other without major temporal delays.

Considering the success factors of agile projects in general, Chow and Cao (2008) conducted a study of the critical success factors in agile development projects. They studied potential success factors in terms of *Quality* (delivering a good working product), *Scope* (meeting all the customer requirements), *Timeliness* (delivering on time) and *Cost* (delivering within estimated cost and effort). The results suggested that the following factors were considered as critically contributing to the success of an agile project in the terms of the abovementioned success attributes. These factors along with their success attributes are as follows.

a) **Correct delivery strategy** in terms of Scope, Timeliness and Cost
b) **Agile software engineering** techniques in terms of Quality and Scope
c) **High team capability** in terms of Timeliness and Cost

Despite the widely recognized role of communication as a factor contributing to the success of development projects, it was not identified as a critical factor contributing to success. However, Chow and Cao (2008) claimed that strong customer involvement in terms of a good customer relationship, strong customer commitment and presence and the customer having full authority plays an important role considering the dimension of Scope. Despite these results, the factors contributing to the success of agile projects appear to be elusive. This elusiveness stems from the fact that the findings reported in (Chow and Cao 2008) reflect a relatively immature state of agile methods. Chow and Cao (2008) postulated that when agile methods become more widely used, it is possible that critical success factors will also change. In addition, the survey does not state whether the agile projects analysed were collocated or distributed. Further, most of the analysed projects (58.7%) involved small teams of up to 10 people. This suggests that most of the data was obtained from small-sized agile efforts.

In 2013, Stankovic et al. (2013) based their work on the study of Chow and Cao (2008) and concluded that customer involvement was not seen as a success factor at all. Whereas Chow and Cao (2008) identified customer involvement as being important in terms of Scope, this study did not confirm this conclusion. In this particular study, most of the people involved were working in distributed environments. Stankovic et al. (2013) used the same set of success factors as Chow and

Cao (2008) and concluded that the nature of the project should be limited to non-life-critical efforts, which contributed to success in terms of Timeliness and Cost. In terms of Cost, limiting the use of agile methods to projects with dynamic and accelerated schedules can be considered as potentially critical success factor. None of the success factors was confirmed by both studies. This inconsistence with the results of the previous study (Chow and Cao 2008) led Stankovic et al. (2013) to a postulation that when assessing the success of an agile project different methods while modelling the success, or different success factors against which project is analysed, should be considered. Further, these results appear to confirm the claim of Chow and Cao (2008) that success factors are elusive and have evolved as agile methods have matured.

2.5 The current state and future of agile methods

According to Dingsøyr et al. (2012) 1551 research papers on agile software development were published between the years 2001 and 2010. Their analysis also indicated that the number of annual publications was steadily increasing until 2009, after which a rather steep decline in the number of publications occurred. The trends of agile literature (total amount, numbers of conference papers and journals) are depicted in Figure 5.

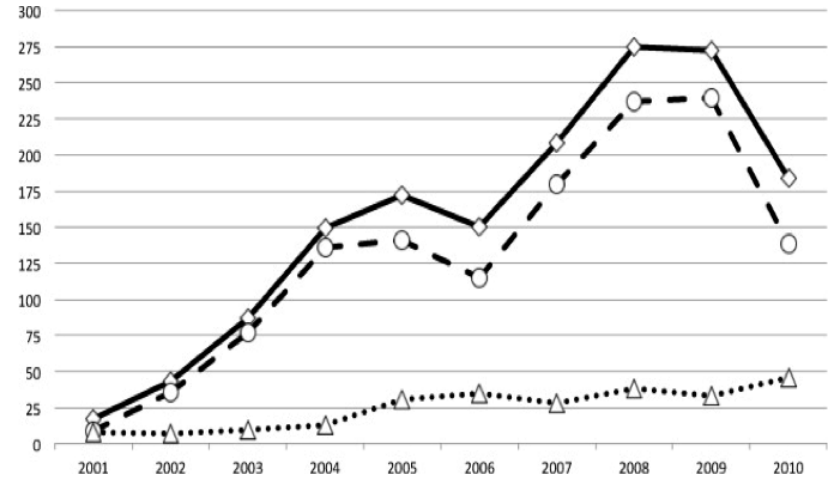

Figure 5[6]. Publications on agile software development ranging from 2001 to 2010, indicating the total number of publications (top curve), the number of conference papers (middle) and journal articles (below) as presented in (Dingsøyr et al. 2012).

Dingsøyr et al. (2012) speculated that the reason behind the steep decline in the number of total and conference publications stems from excluding the 2010 Agile conference from the ISI Web of Science database which they used as the source

[6] Reprinted from Journal of Systems and Software, Vol 85, issue 6, Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N.B., A decade of agile methodologies: Towards explaining agile software development, pp. 1213–1221. Copyright (2012), with permission from Elsevier.

for collection of publication data. The number of journal articles has increased, which Dingsøyr et al. (2012) interpreted as an indication of a maturing field of study. In addition, this research has been conducted in 63 countries in all continents, making agile research a truly global phenomenon, despite the fact that the majority of the studies emerge from the US, Canada and Western Europe. Despite the increasing maturity of the studies, Dingsøyr et al. (2012) called for more attention to theoretical underpinnings of agile methods. According to these authors, theory-driven research should help to separate true innovations from remixes and reinvention of old approaches, and therefore help to adopt innovations faster in the future.

Considering agile methods specifically in the context of GSD, a study conducted by Hanssen et al. (2011) concluded that both globalization and "agilization" are stable trends among the software companies. However, they recognized that there is still a strong need for further studies. Similarly, Jalali and Wohlin (2012) suggested that there are not a sufficient number of studies analysing the challenges of applying agile methods in GSD. Issues are documented from both fields but their combination is not well documented. They concluded that there is a need for further studies in the challenges and benefits of combining GSD and agile methods. "Customising" of agile methods and selective use of agile practices to fit the situations in which they were adopted should therefore lead to determining how much change should be "allowed" for these methods to remain agile in the GSD context. (Jalali & Wohlin 2012).

Agile methods themselves are evolving. For example, applying the concepts of Lean Software Development to agile methods appears to be a phenomenon of interest. Wang et al. (2012) studied the application of lean approaches in agile software development. Wang et al. (2012) labelled such approaches as "Leagile" software development. Their study concluded that there are different ways of applying lean concepts, practices and principles in agile development to serve different purposes. Wang et al. (2012) emphasised that there is no universally applicable single solution for taking this approach. Instead, independent organizations should reflect their own project objectives, development contexts and constraints before taking this approach. However, the means to effectively tailor these approaches to suit the needs of an organization in the context in which it operates is a challenge that should be studied further. (Wang et al. 2012).

Considering the discussion above, it appears that agile methods and their applications and extensions are both in transition and in the process of maturing. It also appears that the implementations of agile methods vary considerably which is indicated by their adaptation to different situations and contexts. Perhaps the realisations of the ideas behind agile methods are as unique as the projects and organizations applying them. The findings of Hansson et al. (2006) might support this postulation. Their study indicated that companies deploy different processes depending on the type of software being developed, and depending on the customer and the size of the project and the company itself. The companies appear to be eminently capable of combining traditional and agile practices according to the needs at hand. (Hansson et al. 2006)

2.6 Summary

Since the inception of agile methods they have gained significant interest. Perhaps due to this and because of the development of several different agile approaches, there are several definitions that approach agility from various perspectives. The concepts behind agile methods are not new, and there are commonalities between agile methods themselves and also between agile and more traditional development approaches, in addition to the differences stemming from their different philosophies in approaching software development. Both scientifically validated and practical industrial results suggest that there are benefits resulting from adopting agile methods. Similarly, there are challenges that appear to focus around organizational factors, such as organizational culture, change resistance towards agile and mismatches between non-agile and agile development approaches. It has also been noted that the original idea of having small collocated teams working closely with the customers does not work well in larger scale settings. One possible solution that could answer this challenge would be to tailor agile methods according to environmental needs and to find a balance between agile methods that provide responsiveness and traditional methods that introduce the discipline that supports large-scale development efforts.

Although there are several claimed benefits in developing software in a globally distributed fashion, these benefits still appear to be more promises and they are only partially realised. However, there are still significant challenges and it appears that GSD is such a complex phenomenon, involving many aspects, that to identify the exact reasons for failure and success in such efforts is very difficult. Whereas these challenges are also present in agile GSD efforts, adopting agile practices appears to provide benefits in GSD. However, there is evidence that the selection of a development approach, be it agile or traditional, does not contribute to the success, or failure, of a GSD project. Adopting agile methods in GSD is increasingly popular and it appears that the viability and the limitations of agile methods are still a poorly understood subject; how they will work in GSD environments remains an open question.

From the scientific perspective, agile methods have attracted significant interest since their emergence and the literature from this field is maturing. There is an emerging trend suggesting that combining the tenets of Lean Software Development with agile methods is the next step in the evolution of agile methods. Software development methods have come a long way since their emergence and they continue to evolve and embrace the changes around them.

3. Communication in software development

This section discusses communication in software development in general and in the context of agile software development. The elements and actors involved in communication are briefly discussed, as well as the distinctions between formal and informal communication. Communication in agile software development in particular is also discussed, and a description of the role of the customer in agile software development is presented. The discussions in this section also involve communication in distributed environments and its challenges and related solution proposals, and the effectiveness of communication from theoretical perspectives. Finally, a summary of the chapter is provided.

3.1 The aim of communication

Rogers (1986, p.199) defined communication as “a process in which participants create and share information with one another in order to reach a mutual understanding”. Whereas this definition refers to communication between individuals, communication between organizations is more about coping with environmental uncertainty (Goldhaber 1993). Similarly, Daft and Lengel (1986) claimed that organizations process information in order to decrease both equivocality and uncertainty. Equivocality means that information has multiple interpretations which are possibly conflicting (Dennis & Valacich 1999), whereas uncertainty refers to lack of information (Daft & Weick 1984). Considering these definitions, the aim of communication is to share information in order to achieve a mutual understanding by reducing uncertainty and equivocality.

3.2 The elements and actors of communication

One of the most prominent studies of the elements involved in communication was presented by Shannon in (1948). In this presentation, a communication system is described as presented in Figure 6.

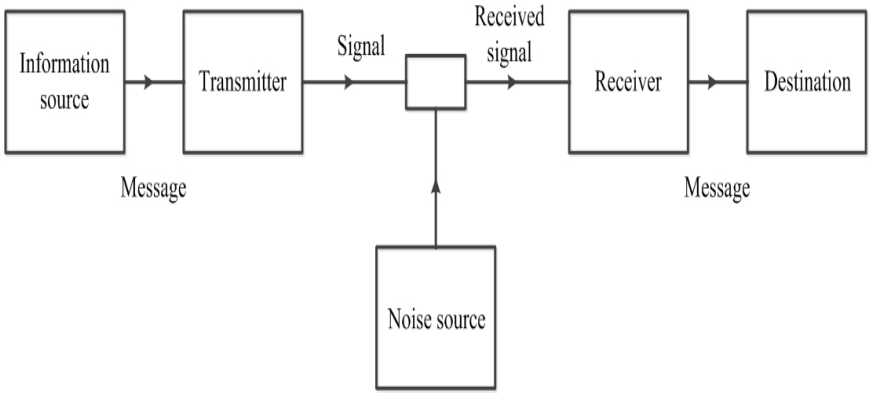

Figure 6. The elements of communication as presented in (Shannon 1948).

According to this model, communication begins with an *information source* which produces a message or a sequence of messages for the system or person for which the message is intended. A *transmitter* operates on the message in order to produce a signal that can be transmitted over the selected communication channel, which is the medium used for conveying the message. The *Receiver* decodes the message from the signal for the use of the *Destination*. Communication can be hindered by *Noise*.

Software development organizations involve several stakeholders that are contributing to development efforts. For example Pikkarainen et al. (2008) defined actors within software development organizations based on the existing literature (Boehm 2003, Malone & Crowston 1994, Leon 1995). In his work, Boehm (2003) identified users, acquirers, developers, managers, designers and maintainers as stakeholders in software development projects. In addition, Boehm (2003) identified sales people and other fulfilment personnel as additional success-critical stakeholders necessarily involved in the development process and the system definition. Leon (1995) also identified stakeholders affecting the development work. These stakeholders include support staff in the organization, such as quality engineers ensuring the quality of the product, and development teams whose work has interfaces with other development teams. Further, Malone and Crowston (1994) identified customers, individual programmers or groups of programmers and managers as actors in information sharing and coordination in their work related to coordination theory. The actors summarized by Pikkarainen et al. (2008) based on the abovementioned literature are as follows:

1. *Software Development Team*. Consists of developers and includes a team leader responsible for conveying project-related information to other stakeholders.
2. *Management*. Responsible for organization-wide strategic decisions.

3. *Customers*. Originally, Pikkarainen et al. (2008) specified customer representation in their work as *External Customers.* They were characterised as "receivers of the developed products" (Pikkarainen et al. 2008, p.311) such as sales people who sell the product, actual customers buying the product or people using the product. However, e.g. Koch (2005) distinguished between *internal* and *external* customers. Internal customers belong to the same organization, i.e. the customer can be another division or a group of people within the organization implementing the software.

4. *Enterprise Staff*. This refers to technical staff in the company who affect and are involved in projects or who will maintain the developed product in the future.

5. *The Support Group*. This refers to a group of stakeholders from different parts of the company and who are responsible for supporting the project. For example, architects or quality engineers belong to this category.

3.3 Formal and informal communication

Communication can be divided into formal and informal communication. Formal communication was defined by Kraut and Streeter (1995) as communication that takes place through structured meetings, writing and through other communication channels that are impersonal and relatively non-interactive. In addition, formal communication channels are prescribed and determined by job functions or organisational hierarchy (Goldhaber 1993). Formal communication channels, such as inspections, specifications and structured meetings are useful for routine coordination activities. (Kraut & Streeter 1995)

Informal communication has been characterised as a peer-oriented, interactive and personal process taking place outside official reporting structures and often without management's knowledge (Kraut & Streeter 1995, Herbsleb & Grinter 1999). In addition, informal communication enables higher flexibility in dealing with ambiguous or uncertain topics, decisions and tasks. Further, informal communication plays a particularly important role e.g. in learning the organization's culture and in forming relationships (Fish et al. 1992). Informal communication also helps with filling in the missing details and rapidly correcting mistakes related to a particular topic (Herbsleb & Grinter 1999). Physical proximity has a pivotal role when considering informal communication. Kraut et al. (1988) studied the impact of physical proximity of researchers on the frequency of informal communication between them. Their findings indicated that physical proximity was strongly related to frequency of communication, manifesting itself as a radical decrease of informal communication when the workspaces were not adjacent. Further, physical proximity has an impact on the quality of communication, which improves when R&D personnel are located closer together. However, the amount of communication did

not increase in the study reported in (Moenaert & Caeldries 1996). Informal communication, such as hallway conversations, workshops and telephone calls, is needed when facing uncertainty or unanticipated problems, which are typical occurrences in software development. In addition, the need for informal communication increases significantly as complexity and size of the software increases. (Kraut & Streeter 1995)

3.4 Communication in agile software development

By its nature, software development is usually work characterised as collaborative, complex and cooperative, involving a multitude of skills, roles and knowledge. The claim made by Šmite (2006) about communication being an integral part of any relationship has been backed up by several authors, stating that effective communication is one of the most essential attributes contributing to the success of any development effort, e.g. (Saeki 1995, Beck 2000, Bostrom & Thomas 1983, Edstrom 1997). Communication also has a pivotal role in almost all collaboration processes and practices (Paasivaara & Lassenius 2003). Communication is generally a time-consuming activity. According to Perry et al. (1994), a significant proportion of the developers' time is spent interacting with other people, predominantly through interpersonal communication.

Communication is by no means an easy task. In order to achieve an effective level of communication, the participants need to share common experience about the subject being communicated so that they can compensate for the gaps in communication. However, it should be accepted that the topics being communicated cannot always be fully understood by the participants (Cockburn 2002). These breakdowns, which appear to be a commonplace phenomenon (Curtis et al. 1988), can have dire consequences. According to Counsell et al. (2005), poor communication often results in severe overruns in budget and schedule as well as inadequately defined requirements.

In traditional, plan-driven development methods communication is formal and relies on explicitly documented information (Boehm & Turner 2003b, Nerur et al. 2005). Furthermore, a characteristic of communication in plan-driven software development is that the communication tends to be unidirectional, passing from one entity to another rather than between them. Items such as progress reports and process descriptions are examples of communication artefacts that are almost always communicated one-way. (Boehm & Turner 2003a)

Agile methods take a different approach to communication between the stakeholders, with emphasis on informal, preferably face-to-face communication and high reliance on tacit knowledge. In agile development, this informal communication is achieved by integrating all the necessary stakeholders closely together. Beck proposed in the context of XP that all the project participants should be collocated in the same shared workspace in order to achieve effective communication (Beck 1999). This includes the customer, who is seen as an integral part of the team; direct communication between the customer and the rest of the team

mitigates misunderstandings and establishes trust between the two parties. (Sillitti et al. 2005)

Pikkarainen et al. (2008) studied the effects of agile practices on communication. Their study approached communication *internally* among the project leaders and developers and *externally* in the interface between the developers and other stakeholders. The analysed agile practices were from XP (Beck 2000) and Scrum (Schwaber & Beedle 2002). Considering the impact of agile practices studied by Pikkarainen et al. (2008) in the field of communication, both beneficial and disadvantageous findings emerged. The findings that improved internal communication are summarized in Table 6.

Table 6. Benefits of agile practices in internal communication (Pikkarainen et al. 2008).

Internal communication	
Agile practice	*Findings*
Open office space	The need for documentation may decrease since everyone in the project has knowledge of the common goals and their status.
Daily meetings	Daily meetings are a good mechanism to keep the developers and the project leader and, in some cases, the customers aware of the status of the project.
Story/task board (information radiator)	Information about the status of the project was available at the wall and this allowed everyone to see the status at one glance.
Iteration planning sessions	The whole team was aware of the project plans and the goals of the next iteration. The use of user stories brought the customers closer to the development and helped reveal the essential requirements.
Iteration retrospectives/reflection workshops	An efficient practice for improving and deploying agile practices.
Pair programming	An efficient mechanism for conducting code reviews. However, seen also as a difficult practise in daily use. This latter viewpoint was from a case in which pair programming was an optional practice.
Continuous integration	Efficient practice for communicating the current status of the project to the people conducting testing. This made testing activities easier.

Based on these findings, agile practices improved communication from multiple perspectives. Agile practices improved awareness of the project as a whole, its status and common goals, and helped to identify the core requirements of the project and to identify programming errors. Further, communication during retrospective sessions helped to improve the work in general. In addition, the need for documentation may decrease. Agile methods do not exclude explicit documentation altogether, although documentation produced in agile projects should be guided by the emphasis on a minimum essential amount (Boehm & Turner 2003a). Similarly, plan-driven methods do not exclude interpersonal communication, which is used as a tool for ensuring consistent shared understanding on the semantics and intents of the documentation. (Boehm & Turner 2003a). Further, Bakalova and Daneva (2011) described a traditional plan-driven project applying customer-intensive communication practices: workshops at the beginning of the project, follow-up workshops during weekly scheduled sessions and frequent meetings also following a scheduled plan. In this case, the communication was active and informal although it was planned and scheduled in advance rather than being spontaneous. (Bakalova & Daneva 2011).

Considering external communication, Pikkarainen et al. (2008) identified the following benefits that are summarized in Table 7.

Table 7. Benefits of agile practices in external communication (Pikkarainen et al. 2008).

External communication	
Agile practice	*Findings*
Iteration planning	A systematic way of sharing information between stakeholders. Improved visibility to short term focus of the project.
Iteration reviews	A systematic way of sharing information between stakeholders. Improved visibility to short term goal/focus of the project.

These practices helped to provide a better understanding of the shorter term goals and focus of the project, since planning and review meetings were conducted regularly at short time intervals. In addition to these benefits, Iteration review practice also helped to understand the product requirements and features.

In addition to benefits, the findings also indicated “hurdles” in agile practices that hindered communication both internally and externally. Table 8 presents the hurdles that were encountered in internal communication.

Table 8. Hindrances of agile practices in internal communication (Pikkarainen et al. 2008).

Internal communication	
Agile practice	*Findings*
Open office space	Noisy environment creates difficulties to focus on the work.
Product backlog	Hindered overall focus of the project. The backlog included a significant number of requirements, and requirements were added to the backlog without proper analysis. In addition, changes in requirements occurred constantly.
Product backlog and iteration planning	Large numbers of requirements were difficult to manage and iteration planning sessions were too short to manage the number of features.

The large number of requirements that were introduced during the project and that were changing affected the longer term focus of the project in a negative way. In addition, the time to manage the requirements in planning sessions was too short. Considering the challenges of Open office space practice, similar findings were reported e.g. by Teasley et al. (2002) They studied physical collocation of developers and customers and applied a "war room" approach which was essentially the same as the shared workspace in agile development. The war room was considered too noisy and developers felt that their work was too closely monitored and that they lacked privacy. Some felt that working in a collocated environment required more concentration. (Teasley et al. 2002) It should be noted that although the study was not focused on agile software development, the development approach contained several agile elements, such as time-boxing and prioritisation, introduced e.g. in Scrum (Schwaber & Beedle 2002, Schwaber 2004). Hence, Open office space can both improve and hinder communication in agile software development. However, Open office space was generally evaluated as a positive practice in the study by Pikkarainen et al. (2008).

Table 9 presents the hindrance of agile practices on communication between developers and other project stakeholders, i.e. in external communication. These stakeholder groups are described in Section 3.2 of this work.

Table 9. Hindrances of agile practices in communication between developers and other stakeholders (Pikkarainen et al. 2008).

External communication	
Stakeholder group	*Findings*
Enterprise staff	Conflicting expectations concerning information sharing. Developers wanted to share information in an interpersonal manner, whereas Enterprise staff expected formal documentation. The principles and the ideology of agile methods were not understood by the Enterprise staff.
Support group	Agile practices did not provide sufficient information in order to conduct testing. The quality engineers responsible for testing expected documentation that would have provided adequate design information. Testing was also difficult because the feature implementation took until the last day of an iteration and the overall testing needed to be completed one day after the end of an iteration.
External customers	Difficulties in backlog management resulted in requirements that could not be understood by the developers. The short time allocated for planning sessions was insufficient to clarify uncertainties. This suggests that time-boxed meetings are not a suitable approach when the complexity of a system increases. The addition of new requirements without proper analysis led to difficulties in backlog management. Daily meetings were seen as too time consuming.
Management	Although anyone from the development team could participate in management level meetings, the management felt that the project leader would be the key person with whom to share information.

The study concluded that all the analysed agile practices helped to improve communication, but also indicated that the practices of XP and Scrum do not provide sufficient communication mechanisms in extended environments involving several stakeholder groups and development teams (Pikkarainen et al. 2008).

3.5 The role of the customer in agile software development

The customer's role in agile development projects is essential and hence creates specific requirements for the customer. In a workshop conducted in 2001 during

the XP2001 conference, van Deursen (2001) listed the following responsibilities for a customer working in XP projects:

1. To understand the customer wishes, maintain regular contact with the end-users and balance their potentially conflicting interests.

2. To talk with the developers, clarify feature requests when needed and understand some of the technical concerns which the developers may encounter.
3. To specify functional tests for user stories and to verify that the tests run correctly.

4. To participate in the iteration planning and release sessions.

5. To maintain good contact with management, explain progress and to justify the time spent with the developers.

Boehm (2002) listed customer attributes that are essential for the success of the project. These qualities: committed, representative, knowledgeable, collaborative and empowered, are also valid for customers involved in agile development projects. Further, Boehm stated that the agile methods reach their full potential when customers with the abovementioned qualities work in a dedicated mode with the developers and when the tacit knowledge which these customers possess is sufficient for the full span of the application (Boehm 2002).

Considering the benefits of close customer collaboration, instant feedback from the customer when it is needed comes naturally from physical collocation. In addition Hanssen and Fægri (2006) conducted a longitudinal case study in which they identified certain benefits stemming from close customer collaboration. They noticed that this close relationship significantly increased the motivation of the developers and also their confidence, since the other stakeholders assisted in prioritizing project goals. Furthermore, the direct collaboration with users increased both the quality of communication and understanding of the real business problems. (Hanssen & Fægri 2006)

However, the role of the onsite customer with constant availability to developers as defined in (Beck 2000) has been identified as challenging. The role is demanding, requiring a strong ability to resolve issues fast (Koskela & Abrahamsson 2004). Martin et al. (2004) studied three projects using XP and concluded that the customers were experiencing stress and worked long hours in order to fulfil the responsibilities of the role. This resulted in a need for making the on-site customer's role more sustainable. Murru et al. for their part claimed in (2003, p. 42) that "XP's most problematic feature is the amount of on-site customer involvement it requires". In addition, the close collaboration itself can create challenges. Hanssen and Fægri (2006) claimed that although strong cooperation is undertaken with a small selection of customers, this reduces the capabilities to capture the needs of other non-appointed customers. This calls for a careful assessment of who should

be the stakeholders with whom collaboration is initiated in order to take the full set of relevant aspects, such as quality drivers, into account while developing the software product. (Hanssen & Fægri 2006)

Sometimes it is not possible for the organizations to utilize the benefits of instant feedback and steering of the work provided by an agile on-site customer. According to Grisham and Perry (2005), only a few XP projects have been able to implement this practice to the full. Further, the practice has been only partially realized through having knowledgeable managers or engineers assuming the role of the customer either part-time or full-time if the customer has been unavailable. In some cases, customers can be simply too valuable for their employers, or the developers and the customers can be too remote from each other to work as was anticipated by XP (Jeffries et al. 2001). Sometimes the customers may even be unwilling to participate actively in the development process (Farell et al. 2002). This practice was later replaced with *Real Customer Involvement* (Beck & Andres 2004), which promotes less active customer participation, for example weekly meetings, instead of constant availability.

Communication with the customers (and other stakeholders) can be conducted via *direct* or *indirect* communication links, as explained by Keil and Carmel in (1995). Direct contact (i.e. direct communication link) between e.g. customers and developers is preferable compared to an indirect link, due to decreased filtering of information or information distortion. Considering information distortion, according to Melnik and Maurer (2004) the information distorts and mutates during passage through a chain of people. The more there are people in the chain, the greater is the possibility of mutation and distortion. Direct links are seen as beneficial especially in situations in which there are high levels of ambiguity, such as in communicating system requirements, for example (Keil & Carmel 1995). When communication is conducted via indirect links, customers and developers, for example, do not communicate directly with each other but the communication is conducted via surrogates or intermediates. Customer surrogates are stakeholders that are not real customers but are treated as such for the purposes of feedback and requirements gathering. Intermediates for their part are stakeholders that are situated between the developers and the customers. (Keil & Carmel 1995). Using intermediates (proxy customers) can create additional challenges, since using proxy customers in detailing the requirements instead of the real customers can often lead to misunderstandings, and further to features that are not implemented as expected by the customer (Therrien 2008).

3.6 The effectiveness of communication from the theoretical perspective

Agile methods prefer face-to-face communication since this is seen as the most efficient way of communication. However, exploring the efficiency of communication media deserves more attention. In the following, the efficiency of communication media is discussed through two different communication theories that were

used in the original publications. These theories are Media Richness Theory (MRT) and Media Synchronicity Theory (MST).

3.6.1 Media Richness Theory

Media Richness Theory appears to be the most prominent communication theory and it suggests that a media's ability to convey information should be aligned with the needs of the task for the best possible performance. Further, this suggests that certain media are better suited to conveying uncertain or ambiguous information (Daft & Lengel 1986, Daft et al. 1987). In the case of uncertainty, obtaining more information is proposed whereas ambiguity requires exchanging subjective views, further definition of the problem and resolutions of disagreements in order to achieve mutual understanding of the topic being communicated. (Daft et al. 1987). Considering the suitability of communication media for different tasks, ambiguous ones should be managed through rich channels whereas less rich media are effective for processing well understood messages and standard data (Daft & Lengel 1986). The richness of a communication media is defined by four variables:

1. **Feedback.** Instant feedback enables rapid correction of errors and asking questions.

2. **Multiple cues.** Different cues such as the tone of voice and body language can be a part of the message.

3. **Language variety.** Refers to the ability to use different languages, for example numerical data that is capable of conveying precise information and natural language that can deliver a broader set of ideas.

4. **Personal focus.** Personal feelings infused in the message can result in more effective conveyance of the message.

MRT's suggestion is that the richest communication channel is capable of providing instant feedback, can transmit multiple cues, has high language variety and is capable of transmitting emotional contents with the message. Media capable of supporting rich communication should be used if information is ambiguous, whereas less rich channels are suitable for conveying well understood messages (Daft & Lengel 1986).

Selecting an appropriate communication media is also affected by the type of communication, that can be either routine or non-routine (Lengel & Daft 1988). The characteristics of non-routine communication are high levels of ambiguity, time pressure and the element of surprise, which makes this type of communication prone to misunderstandings. Often non-routine communication lacks a common frame of reference (i.e. shared understanding) between the communicating participants since the abovementioned characteristics are typified by novel events.

Rich exchange of information (e.g. face-to-face communication) helps to build the shared understanding.

Routine communication for its part is simple, straightforward and logical and lacks the element of surprise. In addition, a common frame of reference has been previously established. For this kind of communication it is beneficial to use media low in richness (Lengel & Daft 1988).

In summary, face-to-face communication, which is a rich communication media, should be a suitable way to communicate topics with high ambiguity. As an example of this, agile requirements are deliberately left vague and open for further discussion when their implementation becomes timely. Face-to-face communication does not however appear to be the best possible media for communicating unambiguous information. Considering the emphasis on face-to-face communication in agile development, MRT does not fully support the claim that face-to-face is always the most efficient way of sharing information.

3.6.2 Media Synchronicity Theory

Media Synchronicity Theory is an extension of MRT aiming towards prediction of communication performance and examining communication media capabilities in various contexts (Dennis et al. 2008). MST approaches communication through two essential processes, conveyance and convergence.

Conveyance is about transmission of sufficient new information to enable the receiver to form and revise an understanding of the content of the information. Conveyance is time consuming and if it is defective, incorrect conclusions will be reached. Convergence on the other hand aims towards shared understanding of the meaning of information and the participants involved in the convergence process need to mutually agree whether or not this mutual understanding has been achieved or can be achieved. Convergence typically involves rapid transmission of smaller amounts of information (e.g. details related to a particular matter) than conveyance and hence requires less individual information processing than conveyance. If convergence is defective, shared understanding of the topic is missing and the participants are unable to move forward with the subject. In order for an individual to perform conveyance or convergence processes, two individual processes must be engaged. The first is information transmission, which includes the preparation of information for transmission, transmission of information through a medium and receiving it from a medium. The second process is information processing, which consists of understanding the meaning of information and integrating this information into a mental model. In information transmission, the focus is on individuals whereas in information processing it lies within individuals. Both the abovementioned processes are required in convergence and conveyance processes in different proportions. (Dennis et al. 2008)

These processes benefit to different degrees from synchronicity, which is defined as "*the ability to support individuals working together at the same time following a shared pattern of coordinated behaviour*" (Dennis et al. 2008, p.576). The definition does not dictate that the participants should be present at the same

physical location as soon they have the tools that enable them to collaborate in real time with the topic at hand and have a shared focus on the matter. Media synchronicity means the capabilities of media to help in achieving synchronicity between the participants. Synchronicity is not always easily achieved. Synchronous communication is necessary for synchronicity, but it is not necessarily sufficient for it since participants can lack a shared focus during communication. In fact, it is possible to use email in synchronous fashion, even though it is less suited for this purpose compared to e.g. telephone communication or instant messaging. (Dennis et al. 2008)

According to MST, using media with lower synchronicity should increase performance for conveyance processes, whereas convergence processes benefit from using media with higher synchronicity (Dennis et al. 2008). Similarly to MRT, MST defines a set of capabilities that different media are able to fulfil to different extents. These capabilities are described in Table 10.

Table 10. The capabilities of media along with their descriptions as described in (Dennis et al. 2008).

Media capability	***Description***
Transmission velocity: The speed at which the message can be delivered to its recipients through a medium.	High transmission velocity improves shared focus between message senders and receivers. This will positively affect the media's capability of supporting synchronicity. High transmission velocity supports *convergence.*
Parallelism: The extent to which the medium can support the transmission of signals from multiple senders simultaneously.	High parallelism lowers shared focus between message senders and receivers. This has a negative impact on a media's capability to support synchronicity. This supports *conveyance*.
Symbol sets: The number of ways that information is allowed to be encoded for communication by media.	a) Media that are able to transmit more natural symbol sets, such as physical, visual, and verbal cues have a better support for synchronicity and therefore *convergence* than media with less natural symbol sets. The latter are more suitable for *conveyance.* b) Using a media that has a symbol set with better suitability to the content of the message will improve information transmission and processing, and therefore will have a better support for synchronicity and for *convergence*.

Rehearsability: The extent to which the media allows the sender to fine tune or rehearse an intended message while it is being encoded, before sending.	High rehearsability lowers the shared focus between message senders and receivers. This will impact negatively on how media is capable of supporting synchronicity. High rehearsability is beneficial in *conveyance*.
Reprocessability: The extent to which a message can be reprocessed or re-examined by the media during encoding, either after the communication event has passed or within its context. Reprocessabiiity is especially important for transmission of large volumes of information and complex or new information.	High reprocessability lowers the shared focus between message senders and receivers and has a negative impact on a media's ability to support synchronicity. High reprocessability supports *conveyance*.

Each media supports these attributes to different extents and it is the combination of these characteristics that determines whether a particular media is capable of supporting synchronicity. Table 11 presents the capabilities of communication media to support synchronicity as discussed in (Dennis et al. 2008).

Table 11. The ability of different media to support synchronicity as described in (Dennis et al. 2008).

Communication media	Ability to support synchronicity
Face-to-face	High
Video conference	High
Teleconference	Medium
Synchronous instant messaging	Medium
Email and asynchronous electronic communication	Low
Voice mail	Low
Fax	Low
Documents	Low

As can be seen, synchronous media are more suitable for convergence than less synchronous media. Agile methods explicitly emphasise the use of media supporting synchronicity and it is claimed that these media, especially face-to-face, are the most efficient means of communication. However, according to MST this should be reconsidered.

MST does not indicate that face-to-face is the most efficient means of communication. MST in fact states that there is no single media that would be inherently better than any other, and in order to complete tasks successfully both conveyance and convergence processes are needed. Different media serve these processes to different extents and hence it is proposed that these media are used

either simultaneously or in succession. Furthermore the context in which the media is used affects its suitability for particular communication situations. This suitability is affected by the communication processes, and individuals themselves engaged in communication in the given social context also have an effect. (Dennis et al. 2008) A claim made by Robert and Dennis (2005) provides insights considering the efficiency of face-to-face communication in agile development. They claimed that complex messages are unlikely to be elaborated using high social presence media (i.e. interactive media with high support for synchronicity). This postulation suggests that face-to-face might not be a suitable media for communicating complex information that is not mutually understood by the participants. In addition, in order to achieve this mutual understanding a media capable of supporting conveyance should be used in order for the recipients to create their individual understanding on the matter, and the missing details and misunderstandings could then later be converged through an appropriate media.

MST also includes Appropriation Factors which indicate that the need for using different media is dynamic and evolves over time. Appropriation Factors influence how the media is used by people and propose that when the familiarity between communicating participants increases both as a result of familiarity with the tasks they are working with and the communication media used, the need for high synchronicity media is reduced. (Dennis et al. 2008).

3.7 Communication challenges and solution proposals in distributed environments: a toolbox

The importance of informal communication is paramount in globally distributed environments. The role of informal communication is critical in rapid dissemination of knowledge, especially when unforeseen events and changes occur. Global distribution has negative impacts on informal communication, and when changes or issues are raised through formal channels this often results in surprises and prevents access to the rationales behind the issues, thus increasing the time needed to get in touch with the relevant stakeholders who have information about the situation. (Bruegge et al. 2006)

Whereas agile methods rely on tacit, interpersonal and informal communication, Lee et al. (2006) emphasized the importance of formal and codified knowledge. They stated that the role of detailed, comprehensive documentation and explicit codified knowledge is essential in global contexts due to difficulties in communication and sharing of tacit knowledge. In global contexts, formal communication is also important due to the fact that it is often more effective than informal communication, which suffers from the negative impacts of language barriers and cultural differences. (Lee et al. 2006)

Communication in distributed environments can be challenging. The barriers related to geographical, temporal and cultural distances (Noll et al. 2010) are identified as the key obstacles to successful communication in globally distributed con-

texts. Further, according to Holmström et al. (2006b), what makes communication in such environments so challenging is the combination of these distances.

Geographical distance makes face-to-face communication difficult and hinders informal communication and idea sharing (Conchúir et al. 2009, Ågerfalk & Fitzgerald 2006, Ågerfalk 2004). Temporal distance for its part reduces opportunities for synchronous communication due to the lack of overlapping working hours (Ågerfalk & Fitzgerald 2006). This may lead to delayed feedback due to delays in responses and increases the time for resolving problems at hand (Ågerfalk 2004, Boland & Fitzgerald 2004). If overlapping work hours are arranged, this may lead to unconventional communication hours, which is consuming (Holmström et al. 2006a, Sarker & Sahay 2004, Conchúir et al. 2009). Cultural distance can increase misunderstandings in communication (Holmström et al. 2006a, Conchúir et al. 2009, Summers 2008, Ågerfalk & Fitzgerald 2006, MacGregor et al. 2005). Disagreements and negative issues are also not necessary willingly expressed due cultural inhibitions. (Lee & Yong 2010, Drummond & Francis 2008). In addition, language barriers have a negative effect on communication (Layman et al. 2006, Uy & Ioannou 2008, Kajko-Mattsson et al. 2010).

In addition to geographical, cultural and temporal distances, other factors can also have a negative impact on communication between the distributed partners. A lack of face-to-face communication and direct communication hinders the building of trust between distributed partners (Conchúir et al. 2009, Therrien 2008, Lee & Yong 2010) and the often proposed solution of using communication tools for mitigating the challenges resulting from distance can become a challenge in itself. Issues with communication tools appear to be a common phenomenon in distributed development (Therrien 2008, Ågerfalk 2004, Paasivaara et al. 2008a, Williams & Stout 2008). A lack of active customer involvement, which is crucial in agile development, may also result in a lack of direction of the project. This is particularly troublesome if teams are lacking knowledge considering the domain (Williams & Stout 2008).

Proposals for tackling the abovementioned communication challenges have been provided in the literature discussing distributed software development, both in general and in agile software development. Figure 7 presents the challenges identified in the literature together with solution proposals. Figure 7 hence presents a “toolbox” that can be utilized in order to tackle communication-related challenges. Solution proposals are discussed in more detail.

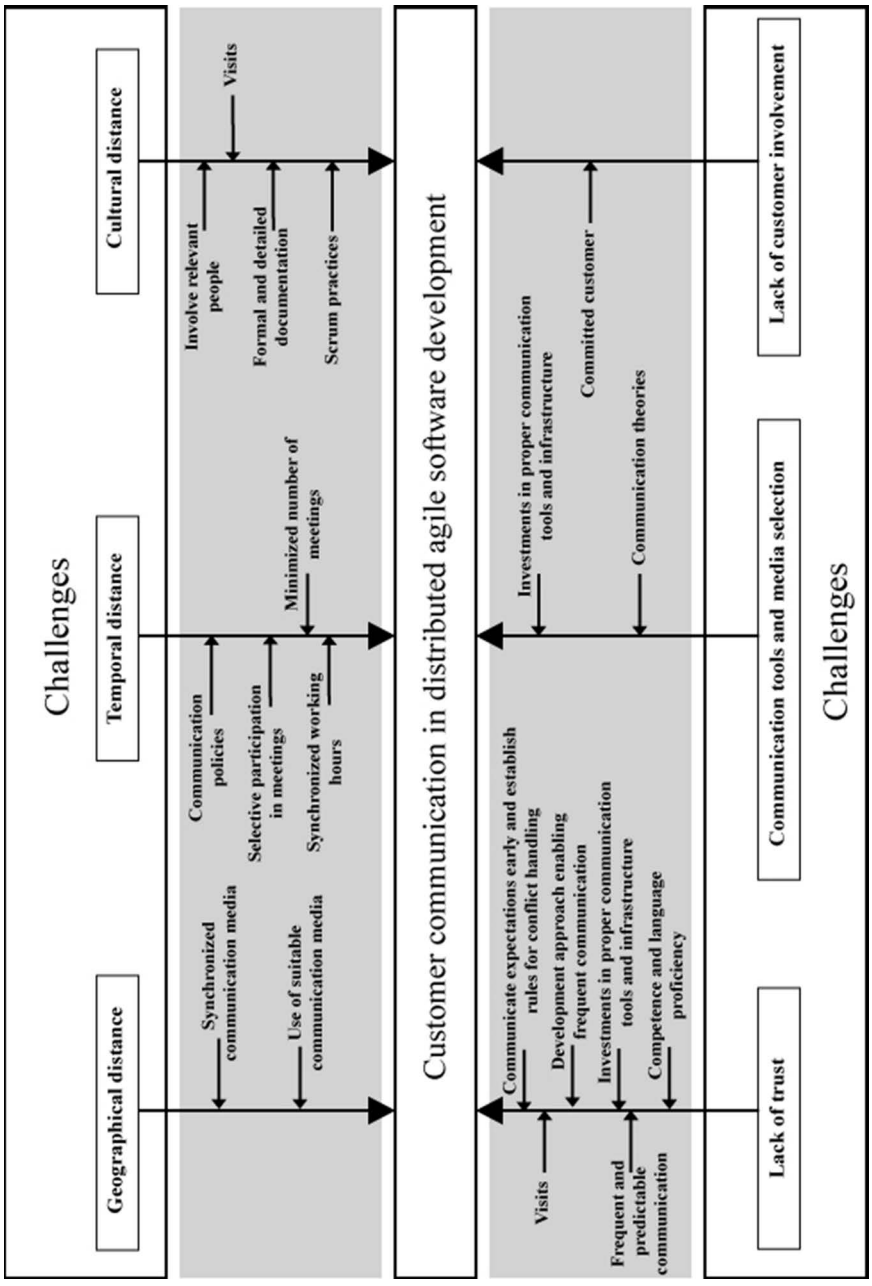

Figure 7. Toolbox for mitigating communication challenges in distributed agile software development.

The solution proposals for geographical challenges are focused on communication tools. This challenge can be mitigated by using synchronous communication tools, such as videoconferencing (Kircher et al. 2001, Sureshchandra & Shrinivasavadhani 2008), whiteboard software (Layman et al. 2006), Web conferencing tools (Danait 2005) and Instant Messaging tools (Danait 2005). In addition to synchronous tools, asynchronous media can also be used for mitigating geographical distance (Bannerman et al. 2012). Issues related to communication tools can be eased by appropriate investments in tools and communication infrastructure. (Ebert & De Neve 2001, Therrien 2008, Kussmaul et al. 2004)

Communication theories can also assist in selecting appropriate media for efficient communication and hence mitigate the challenges experienced in distributed agile software development projects. MRT proposes that rich media, such as face-to-face, are an efficient mechanism for communicating ambiguous topics, and that uncertain matter should be communicated via less rich media. MST suggests that communication media should be selected based on the communication needs, i.e. whether conveyance or convergence is needed in communication. Further, this need evolves over time when communicating participants become more familiar with the each other, their work and the communication media (Dennis et al. 2008). However, the existing solution proposals for communication in distributed environments appear to emphasize the use of rich communication channels that support synchronicity. From the perspective of MRT this would be appropriate in the context of agile methods but considering MST this approach would not necessarily provide the most efficient means of communication in the context of ambiguous matters. Hence, the recommendations for using communication media appear to be somewhat contradictory from the perspectives of the selected communication theories.

For temporal distance, the mitigation solutions are related to applying different practices. Working hours should be synchronized in order to create overlapping time (Bannerman et al. 2012). Since this might lead to unconventional work times, overlapping hours should be kept sustainable (Therrien 2008). In addition, policies concerning participation in meetings can help to keep the work effort sustainable. It has been suggested that the need for meetings requiring all the stakeholders should be minimized (Bannerman et al. 2012) and only key project members should participate in early morning or late evening distributed meetings, rather than the whole team (Williams & Stout 2008). Strict communication policies to reduce communication delays caused by distribution can be applied. For example, emails should be replied to within 12 business hours. (Vax & Michaud 2008)

Tackling cultural distance involves relying on experienced people. For example, experienced domain experts should communicate with distributed teams daily. This should mitigate communication risks emerging from cultural differences by keeping the potential problems transparent. (Summers 2008) Layman et al. (2006) recommend that if project management and development are separated, a role with the purpose of working closely with both development and management should be established. Ideally, the person entrusted with this role should speak all the languages involved in the project in order to deal with the language barriers. In

order to increase the awareness of cultural diversity and how to cope with it, managers should be rotated across cultures and locations. (Ebert & De Neve 2001) Further, regular visits from key stakeholders such as senior developers, managers and business analysts from the customer organization can help the customer organization understand the cultural aspects of the remote site (Sureshchandra & Shrinivasavadhani 2008).

Visits also help with establishing trust between distributed partners (Therrien 2008), since in order to maintain and strengthen trust it is important that members of the teams socialize, and face-to-face communication helps with socialization. (Moe & Šmite 2008). Teams that are experiencing low trust need frequent and predictable communication if trust is to grow (Moe & Šmite 2008). However, too much communication can cause a decline in trust since team members can experience that they are being monitored (Järvenpää et al. 2004). Further, Moe and Šmite (2008), proposed recommendations related to communication for improving trust in distributed environments. These recommendations are derived from the existing literature reported in the study and from the findings of the study itself (Moe & Šmite 2008). These practices are investments in several face-to-face meetings similar to the visits recommended by Therrien (2008), communicating expectations early and establishing rules for conflict handling, ensuring that the team possesses the necessary language skills and competence in order to improve communication and using an adaptive and flexible development method using frequent communication in order to coordinate the work by constant feedback. In this approach, the balance between agility and rigour should be taken into account. Investments should also be made in groupware and team intranet in order to enable efficient communication and compensate for the lack of face-to-face communication. Hence, communication tools and infrastructure can be used to improve trust. (Moe & Šmite 2008). This latter finding is similar to the recommendation for investing in proper communication tools and infrastructure (Ebert & De Neve 2001, Therrien 2008, Kussmaul et al. 2004).

Agile practices themselves also help to mitigate cultural challenges. The Scrum meetings (planning, review, daily scrums, retrospectives) themselves enforce the common work practices on the participants. The meetings also serve as frequent check-points for ensuring that collaboration is maintained and shared understanding remains throughout the project. These meetings should be recorded for later review. (Bannerman et al. 2012). In addition, on the basis of the findings of Lee et al. (2006), formal and detailed documentation can help coping with cultural distance and language barriers.

In order to ensure crucial customer involvement and, hence, active customer communication the customer should be identified before the project begins. This customer contact person must be able to make conclusive decisions concerning the project's scope and functionality, must have a vested interest in the project and needs to be readily available. (Layman et al. 2006).

3.8 Summary

Informal and preferably face-to-face communication has been promoted in agile development as the most efficient communication mechanism. However, in globally distributed endeavours face-to-face interactions can be extremely difficult due to geographical distance, which in turn can be significant. In order to follow the agile tenet of interactive communication, tools that enable it have been proposed to solve the problem. From the theoretical perspective (Dennis et al. 2008), interactive (i.e. high synchronicity) communication is an appropriate way of exchanging information when all the communicating participants have a shared understanding of the topic. For example, agreeing on small details can be conducted via interactive communication channels.

However, if the information is new and complex and the participants do not share a common understanding of its meaning, interactive communication may not be the best option. In such situations, the theoretical proposition is to use media with lesser synchronicity, such as documentation, to convey the information. After the participants have developed their individual understanding on the information it can be converged (potential misunderstandings corrected) using high synchronicity communication media. This suggests that the role of documentation is important even in agile development. In the context of globally distributed agile efforts the role of communication is even more pivotal when considering the suggestions made by Lee et al. (2006).

From the perspective of MRT, face-to-face (i.e. high synchronicity medium) would be a suitable way to communicate ambiguous non-routine topics. This is more in line with the agile methods' recommendation of face-to-face communication that was suggested by MST. Hence, the selection of a communication tool to best serve the purpose based on these theories is contradictory.

The literature survey concerning the challenges conducted for this thesis identified six different challenge areas that can represent a threat to customer communication. These challenge areas include temporal, geographical and cultural distances between distributed partners and issues with lack of trust, communication media and lack of customer involvement. Practices for mitigating their effects have been proposed in the existing literature.

4. Research design

This section discusses the research approaches taken in this study, data collection mechanisms and the data analysis methods applied in the case projects contributing to this dissertation.

4.1 Research approach

This section discusses the research methods and approaches taken in this study. Action research, different philosophical stances that can be taken during case studies and other classifications are also presented.

4.1.1 Action research

During the first case project Action Research (AR) (Susman & Evered 1978) was applied. According to Cunningham (1997, p. 403), the purpose of AR is to "develop concepts which help to facilitate the process of change", and theory "emerges in the process of changing". According to Davison et al. (2004), AR's application focuses on solving organizational problems and at the same time providing contributions to knowledge. According to (Easterbrook et al. 2008) most empirical methods aim towards observing the world as it exists, whereas action researchers have an explicit purpose of improving the current situation by deliberate intervention.

AR is an iterative process that includes the following steps: diagnosing, action planning, action taking, evaluation, and specifying learning (Susman & Evered 1978). In the *diagnosing* phase the problem is identified or defined and *Action planning* focuses on considering the means and alternative courses of actions that could solve the particular problem. In the *Action taking* phase, the course of action is selected. The consequences of actions are evaluated in the *evaluation* phase and the general findings are identified during *specifying learning*. The process is then repeated.

According to Baskerville (1999), AR is not a single, monolithic research method but instead refers to a class of research approaches. Lau (1999) identified three other streams of AR in addition to traditional AR. In Participatory Action Research, the practitioners are involved in the study both as subjects and co-researchers. In

addition, the practitioners "solve problems themselves by setting their own research agenda, collecting and analyzing the data, and controlling overuse of the findings." (Lau 1999, p. 150). Action Science puts emphasis on studying the "participants' behaviours as theories-in-use versus their beliefs as espoused theories" (Lau 1999, p. 150). Action Learning (Lau 1999) advocates programmed instructions, group participation, spontaneous questioning, real actions and learning in different organizational and social contexts.

4.1.2 Case studies

The research during this work was conducted as a series of case studies. Case studies have proved to be useful in situations in which the target is to understand a contemporary phenomenon in complex, real world settings, especially when the boundaries between the context and the phenomenon are not clear. (Yin 2003, Eisenhardt 1989) Research in software engineering has a result-oriented and pragmatic view of research methods rather than a philosophical view (Seaman 2002). However, case studies can be categorized on the basis of the philosophical stance taken in the study. An interpretive approach was specifically taken while conducting the study reported in paper V. In the other papers, philosophical approaches were not explicitly adopted.

Positivistic studies measure variables, seek evidence for formal propositions, draw inferences from a sample to a stated population and test hypotheses. Positivistic studies assume that *a priori* fixed relationships within the phenomena exist, and structured instrumentation is typically used to investigate these phenomena and to serve as an attempt to increase predictive understanding of phenomena by testing theories. (Orlikowski & Baroudi 1991). *Critical* studies aim towards social critique and towards identifying the alienating and restrictive current conditions and making them visible (Klein & Myers 1999). *Pragmatism* acknowledges that all knowledge is incomplete and approximate and that its value depends on the methods used to obtain the knowledge (Menand 1997). According to Easterbrook et al. (2008), pragmatists approach the judgement of knowledge on the basis of how useful it is in solving practical problems, i.e. "truth is whatever works at the time" (Easterbrook et al. 2008, p. 292). In addition, truth is relative to the observer, since what is perceived useful by one person might not be useful to another. In order to overcome criticism, the importance of consensus is emphasised: rational discussion uncovers the truth. Easterbrook et al. (2008) also claimed that pragmatism is less dogmatic than the other stances, since all research methodologies should be free to be used in order to study a particular research problem.

Interpretative studies aim to understand phenomena through how the participants interpret their context (Orlikowski & Baroudi 1991). Further, generalization is not sought and Orlikowski and Baroudi (1991, p. 5) labelled interpretive case studies as those in which the intent of the research is to *"increase understanding of the phenomenon within cultural and contextual situations; where the phenomenon of interest was examined in its natural setting and from the perspective of the*

participants; and where researchers did not impose their outsiders' a priori understanding on the situation".

Klein and Myers (1999) provided guidelines on how to conduct interpretive case studies. These principles along with their description and application in this study are described in Table 12. Whereas Klein and Myers (1999) did not advocate arbitrary use of just some of the principles while ignoring the others, they stated that the principles are not mandatory rules of conduct, but that discretion should be exercised in deciding "whether, how, and which of the principles should be applied and appropriated in any given research project." Klein and Myers (1999, p. 71).

Table 12. The principles for conducting interpretive case studies and their descriptions based on (Klein & Myers 1999) and how these principles were applied in the research paper V.

Principle	***Description***	***Applied in the case reported in paper V.***
1. The fundamental principle of the hermeneutic circle	Understanding is gained by iterating between the interdependent meaning of parts and the whole that they form.	The data was obtained in iterative fashion using various data sources during the period when the case was conducted. The data was iteratively analysed and used to steer the research further, if that was considered relevant.
2. Contextualization	The subject matter should be set in its social and historical context so that it can be seen how the current situation under investigation emerged. Interpretive research argues that organizations are not static and that relationships between organizations, people and technology are constantly changing.	Customer communication in this study was a non-static, changing phenomenon. Whereas there were defined sessions for customer communication, informal customer communication occurred spontaneously.
3. Interaction	Facts should be produced as a part and parcel of the social interaction between the researchers and participants.	The facts were produced during interactions between the author of the paper and the participants of the study.
4. Dialogical reasoning	Sensitivity is required to possible contradictions between the actual findings and theoretical preconceptions guiding the research	No contradictions were identified.

	design.	
5. Abstraction and generalisation	Details revealed by the data interpretation should be related to theoretical general concepts through application of principles one and two.	Principles one and two were applied in this study as described. Further, the data was reflected against the propositions of the selected communication theory (MST)
6. Multiple interpretations	The researcher should examine the influences that the social context has on the actions under study. Multiple viewpoints along with the reasons for them should be sought out and documented.	The context in which the participants operated was taken into account. Participants' viewpoints were documented and compared.
7. Suspicion	Requires sensitivity to possible "biases" and systematic "distortions" in the narratives collected from the participants.	Critical approach in the data analysis was applied. The possibility that the participants' viewpoints could be biased was recognized during the data collection.

Studies can also be classified by their intended goals. For example Robson (2002) classified case studies as follows:

- *Exploratory* case studies aim towards understanding what is happening, seeking new insights and generating hypotheses and ideas for future research.
- *Descriptive* case studies portray a particular phenomenon or situation.
- *Explanatory* case studies seek an explanation of a problem or a situation, mostly but not necessarily in the form of causal relationships.
- *Improving* case studies try to improve some aspect of the phenomenon being studied.

In order to conduct a study, the unit of analysis needs to be defined. The unit of analysis can be an individual person, a group of people, a product or a policy or a particular role in the organisation, to mention just a few. (Runeson & Höst 2009). In this study, the unit of analysis was a either a single or multiple software development project, depending on the intentions of the study.

Table 13 summarises the case studies. The table presents the number of units of analysis, the approach taken in the study in terms of epistemological stance or the goal of the study and the levels of distribution within the projects.

Table 13. The description of case studies reported in individual research papers.

Paper	*Unit of analysis*	*Study approach*	*Distribution in the case projects*
I	One agile software development project.	Improving study. The presented process was improved from its previous version based on the empirical findings.	One collocated agile software development project.
II	Four agile software development projects.	Explanatory study. The aim of the study was to explain what happens to the software quality at different levels of customer involvement and, hence, communication.	Two collocated agile development projects. Two projects with two distributed development teams. Customers located with one of the teams.
III	Two agile software development projects	Improving study. The aim of the study was to describe the use of existing communication practices and to further increase knowledge about their use. In addition, a new recommendation emerged.	Each project had two distributed development teams. Customers were located with one of the teams.
IV	One agile software development project.	Exploratory study. The aim of the study was to analyse customer communication in a situation in which both traditional and agile methods were used. In order to answer to the challenges, a set of new recommendations emerged.	Two different globally distributed teams. The customer was located with one of the teams.
V	One agile software development project.	An interpretive stance was explicitly taken. The case was exploratory, since the aim was to understand and mitigate communication challenges through the concept of waste and, hence, to seek new insights.	Three different globally distributed teams. The customer was located with one of the teams.

4.2 Data collection

According to Lethbridge et al. (2005) it is important to obtain reliable and accurate information about the phenomenon that is being studied. Surveys and interviews are the most straightforward data collection instruments (Lethbridge et al. 2005),

but in order to improve the validity of the study, several data collection methods should applied as proposed e.g. in (Eisenhardt 1989, Yin 1994, Stake 1995). Lethbridge et al. (2005) divided data collection techniques into three categories that were defined based on the human contact they require. First degree techniques require direct access to participants, whereas second degree techniques require access to the participants' work environment while they are working, but do not require direct access to the participants themselves. Third degree techniques require access to work artefacts only. The second and third degrees differ from each other in that the second degree requires data acquisition while the work is being conducted whereas the third degree has no requirements with respect to when the data is collected.

The data was collected during the studies mainly through first degree data collection techniques. Interviews were the most important data collection technique and all the interviews conducted were transcribed verbatim. The data collection techniques are presented in Table 14 based on the categorization by Lethbridge et al. (2005).

Table 14. The data collection techniques of the original publications following the categorization presented in (Lethbridge et al. 2005).

Original paper	***Data collection techniques***
Paper I	*First degree* • One group interview with the developers after the project ended. • Research diary documenting the observations, research activities, ideas considering the study at hand and future research possibilities etc., • On-site observations. On-site participation during iteration planning, sprint review and retrospective sessions. Participation during the development iterations approximately twice a week.
Paper II	*First degree* • Four group interviews with the developers • Research diary documenting the observations, research activities, ideas considering the study at hand and future research possibilities. • On-site observations. On-site participation during iteration planning, sprint review and retro-

	spective sessions. Participation during the development iterations approximately twice a week. • Email correspondence between the teams and the customers. The researchers were included in this correspondence, but did not intervene in product development related matters. *Third degree* • Quantitative defect data recorded by developers.
Paper III	*First degree* • Two group interviews with the developers • One semi-structured interview with an experienced developer working in one of the case projects in order to cover issues not discussed in the group interviews. This person was the most easily available for the interview. • Research diary documenting the observations, research activities, ideas considering the study at hand and future research possibilities. • On-site observations. On-site participation during iteration planning, sprint review and retrospective sessions. Participation during the development iterations twice a week. • Email correspondence between the development team and the customer during the project. The researchers were included in this correspondence, but did not intervene in product development related matters.
Paper IV	*First degree* • One group interview involving the project manager, a software architect and one developer from

	the other development unit. The U.S. based customer organization was not available for interview.
Paper V	*First degree* • 12 interviews. (9 individual interviews, 3 group interviews). • One group interview focusing on the use of documentation and tools for sharing and storing it. General level development-related matters were also discussed. • 11 observation sessions taking place either onsite or via telephone and screen sharing software. • Research diary documenting the observations, research activities, ideas considering the study at hand and future research possibilities. • Informal discussions with the project manager and case company management about the project. • Email discussions with the case project's project manager. *Third degree* • A document related to how the functionalities were distributed between the sites • A document explaining the requirements creation and backlog refinement processes.

4.3 Data analysis

The data was analysed following the guidelines of Miles and Huberman (1994). During each individual study the initial codes (seed categories) were defined on the basis of the research objective, i.e. what was to be analysed. These codes were adjusted during the studies as considered necessary and additional ones were added as in Thematic Analysis (Braun & Clarke 2006). The interview tran-

scriptions were read through and the findings related to defined codes were grouped into a data display that formed a condensed set of information, i.e. the data reduction process was followed. (Miles & Huberman 1994).

The data analysis in the case project described in paper I was conducted following the steps of AR. At first during the *diagnosis* phase, the practical problem of not having an onsite customer was defined as the problem to be solved. At the beginning of each iteration a set of practices to improve customer communication and involvement was defined. These practices were implemented during *action taking* and their effects were *evaluated* together with developers in post-iteration workshops (i.e. retrospectives). Finally, these *learnings* were incorporated in future iterations. After the project had ended, a single group interview was conducted in order to gain additional insights into the improved version of the process.

In paper II the focus was on collecting quantitative defect-related data. Each project documented the requirements in the form of user stories which were further refined into tasks, as described in XP (Beck 2000). Defects were also documented as tasks. The defect-related tasks were analysed from each project and the type of the defect was classified as either *customer dependent* or *customer independent*. The proportion of defect-fixing tasks from all the tasks as well as the times required to fix them were also extracted. Since it was possible to extract accurate defect data from just one case project, statistical methods were not applied. Instead, the study was able to demonstrate trends on defects within projects with various levels of customer involvement and communication. The interview data was complemented with other research material described in Table 14. All data was analyzed manually.

5. Research contributions

In this section, each individual study included in this thesis is described. This description includes an overall view of the papers and their key contributions.

5.1 Paper I: Extreme Programming: Reassessing the Requirements Management Process for an Offsite Customer

This paper approached customer communication using the assumption that it is not possible to have an onsite customer. In this article, a process for integrating the customer incapable of on-site participation was refined from its previous version. The paper approached customer communication from the perspective of requirements. Despite this angle, the proposed process aims towards customer integration and communication throughout the development project. The case project applied an agile approach called Mobile-D, (Abrahamsson et al. 2004), specifically designed for the development of mobile applications. Mobile-D incorporates all the XP practices, excluding the on-site customer. The iteration length in Mobile-D is one week and the process includes 5 releases. The first and the last release last for one week, whereas the duration of the remaining three is two weeks.

Based on the case results, the paper presents a process and practices that should ensure efficient communication between the customer and the developers, and the customer's involvement in all phases of the project.

The key results of this paper are as follows:

- ✓ A process for integrating customer to an agile development process
- ✓ A set of practices that aims towards this integration and to ensure that customer communication is effective throughout the project.

This paper contributes to research question Q2: How is it possible to involve the customer in the development process in distributed agile development in order to ensure communication and feedback?

5.2 Paper II: A Case Study on the Impacts of Customer Communication on Defects in Agile Software Development

The basis of this publication was the assumption that active customer communication is paramount in agile software development (Beck 2000). The aim of the paper was to study the impacts of various levels of customer communication on software quality. Similarly to paper I, Mobile-D was the development method used in the case projects involved. In this study, four case projects with various levels of customer involvement were analysed. One project had an on-site customer and in the others the customer was available in various degrees.

The results indicated that when the customer participation and, hence, communication decreases, the number of defects that could possibly have been avoided with more frequent customer communication and feedback increases accordingly.

The key findings of this paper were as follows:

- ✓ Increased reliance on less interactive media and lesser customer involvement increases the number of defects.

- ✓ Attention should be paid to selection of suitable communication media to be used during iterations when the customer is not available.

This paper contributes to answering research question Q1: Why is customer communication important in distributed agile software development?

5.3 Paper III: Communication in Distributed Agile Development: A Case Study

This study approached customer communication from the perspective of practices described in (Layman et al. 2006). These practices aim to create an environment that fosters communication in globally distributed agile development projects. In this study, these suggestions were evaluated against empirical findings derived from two different case projects. These projects used Mobile-D as the development approach.

The results indicated that the practices suggested by Layman et al. (2006) are worth considering in distributed agile development, but with some caution. In addition, direct communication, in this case between the developers, is encouraged and added as an additional practice. In this paper, customer communication was not operative during one of the case projects, which caused significant challenges for the development. The team that would have required the customer's input and feedback did not receive steering from the customer and had to rely on their perceptions about the functionalities and how they should be implemented so that they could meet the customer's expectations. The team was not allowed to directly

contact the other team, located at the same site with customer. Instead, all communication was conducted via the customer.

The key findings of this paper were as follows:

- ✓ The studied communication practices should be considered in distributed agile development, but with some caution. An additional practice proposing direct communication between the distributed teams should be encouraged.

- ✓ Inoperative customer communication can cause significant challenges even within small-scale distributed agile projects.

- ✓ Customer relationship should be given extra effort while planning, managing and executing agile projects.

This paper contributes to research questions Q1: Why is customer communication important in distributed agile software development? and Q3: What are the means, practices and tools for improving customer communication in distributed agile development?

5.4 Paper IV: Combining Agile and Traditional: Customer Communication in Distributed Environment

The aim of the study was to analyse communication in the context in which two different methods are being used in a single project. This practitioner-oriented book chapter describes the challenges in communication encountered in a joint development effort between two business units of a single organization. One of the units was using a traditional plan-driven method and the other one used an agile approach. The traditional organization had assumed the role of a customer and provided the agile organization with the requirements they were supposed to implement. The customer organization was located in the United States of America while the other unit resided in Ireland.

The results suggest that it can be difficult for an agile organization to obtain relevant information from a traditional type of business unit working as the customer, even though communication was indicated to be active and it was conducted via multiple different communication media. There were several reasons that contributed to what was referred to as the “information blackout” in the publication. Similar findings are also reported in paper III. In order to overcome the issues reported in this publication, a set of practices was proposed in order to create an environment that enables efficient customer communication.

The key results of this paper were as follows:

- ✓ The combination of a customer who is not participating actively in the development, bureaucratic organizational environment, differences between traditional and agile methods and lack of trust can make distributed agile development very challenging.
- ✓ A set of guidelines aiming towards ensuring that a distributed agile development project has an involved customer who will provide the necessary information to the development teams. These guidelines are presented in the form of questions and are as follows:

    - What kind of information do we need from the other party?
    - Who will be providing this information?
    - Are we able to get this information when it is needed?
    - Are the sources of information committed to provide the information when agreed?
    - Is there something that prevents us from getting this information?

This paper contributes to research questions Q1: Why is customer communication important in distributed agile software development? and Q2: How is it possible to involve the customer in the development process in distributed agile development in order to ensure communication and feedback?

5.5 Paper V: Waste Identification as the Means for Improving Communication in Globally Distributed Agile Software Development

According to Ikonen et al. (2010), applying the principles of Lean production to software development contexts appears to be one of the emerging trends in software development. This journal article approached communication (including customer communication) in globally distributed agile software development from the perspective of involving the concept of waste (i.e. resource-consuming, non-value adding elements) (Womack & Jones 1996) in agile development in order to identify non-value producing elements from communication.

Paper V aimed towards identifying communication-specific wastes. The study identified five wastes that are specific to communication and also indicated that already identified wastes described in (Poppendieck & Poppendieck 2007, Mandić et al. 2010) are also valid in the context of communication. In addition to five identified wastes of communication, this study provided another unique contribution in the form of a process through which communication can be analysed, wastes extracted and appropriate measures for mitigating them defined. In addition to these findings, the product owner's mandatory participation in different development process phases ensured efficient communication.

The key results of this journal article were as follows:

- ✓ Five wastes of communication: lack of involvement, lack of shared understanding, outdated information, restricted access to information and scattered information.
- ✓ Mandatory customer participation during the phases of the development project ensures efficient communication by systematically involving the customer in the process.
- ✓ A process for analysing and improving communication in agile development projects.

This paper contributes to answering research questions Q2: How is it possible to involve the customer in the development process in distributed agile development in order to ensure communication and feedback? and Q3: What are the means, practices and tools for improving customer communication in distributed agile development?

5.6 Summary of contributions

Figure 8 presents the findings in the context of the toolbox depicted in Figure 7, hence complementing the toolbox with the findings of this study. Elements added to the toolbox as the results of this study are highlighted.

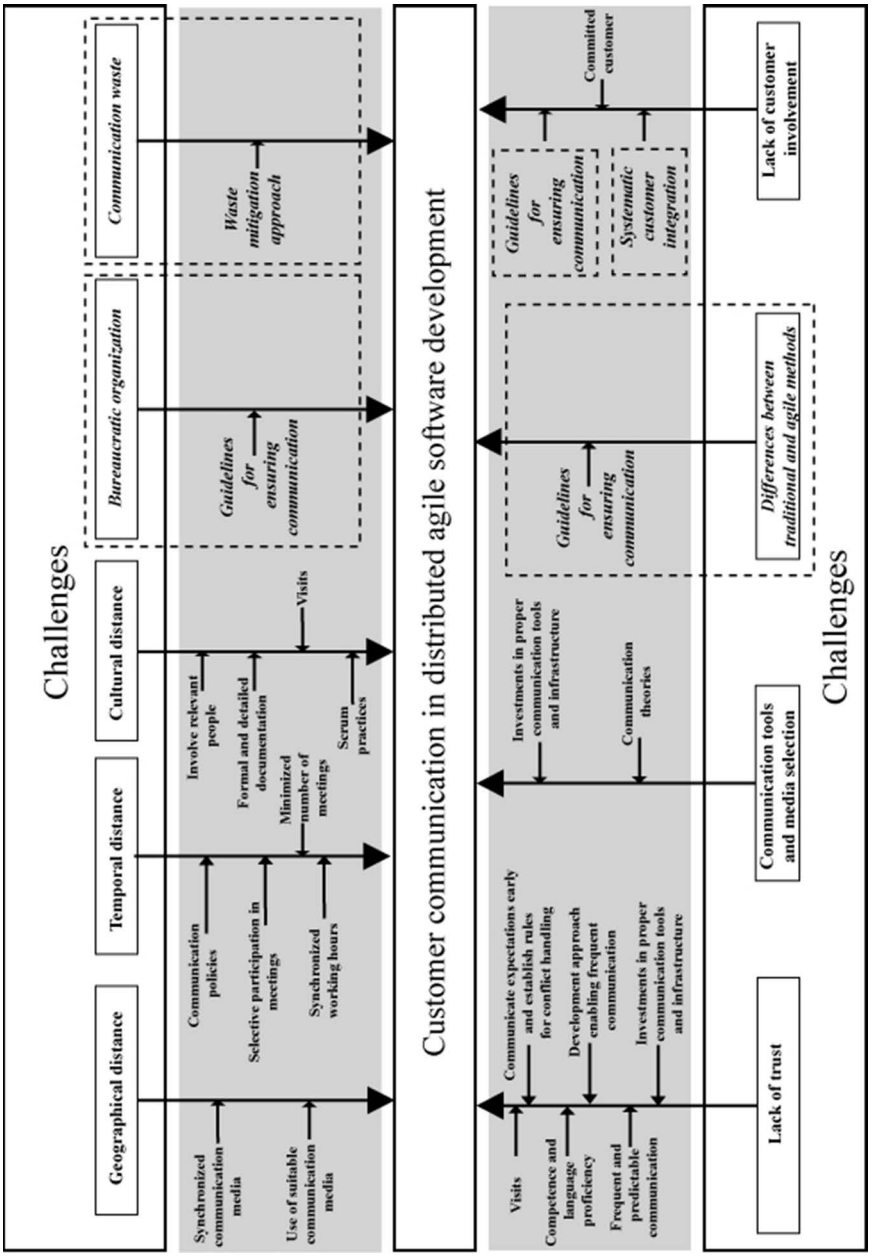

Figure 8. Toolbox for mitigating communication challenges in distributed agile software development complemented with the findings of this study.

This study identified new challenge areas that can hinder customer communication in distributed agile software development. These challenge areas are: bureaucratic organization (paper IV), differences between agile and traditional methods (paper IV) and communication waste (paper V). This study also identified solution proposals for different challenge areas. Challenges related to bureaucratic organization aspects can be mitigated by the guidelines discussed in paper IV, and these guidelines can also be applied in the context of differences between agile and traditional methods. A systematic waste mitigation approach in order to identify communication waste and to improve communication in distributed agile development projects was presented in paper V. This study also provides additional solutions to the challenges stemming from the lack of customer involvement. Paper I provides an approach for integrating the customer to an agile development project, and the guidelines presented in paper IV and the finding related to mandatory customer participation throughout the project reported in paper V provide means to involve the customer in the process.

6. Discussion

In the following, both pragmatic and theoretical implications of this thesis are discussed on the basis of the main results. The main results are the toolbox for identifying and mitigating customer communication challenges in distributed agile software development described in Section 5.6, the new concept of communication waste and the implication that lack of trust represents potentially the single most important factor threatening communication in distributed agile developments. These implications are discussed in the following sections.

6.1 Implications for research

In this section, the main implications of this study for research are presented. In addition, the findings related to both Media Richness Theory and Media Synchronicity Theory are discussed.

As a unique contribution to the field of communication in agile development, this thesis introduced the *five wastes of communication*; lack of involvement, lack of shared understanding, outdated information, restricted access to information and scattered information. Waste is a concept emerging from Lean production and refers to actions that consume resources but do not add value (Womack & Jones 1996). This concept has been adapted to software development, e.g. by Poppendieck and Poppendieck (2007) and Mandić et al. (2010). These authors presented their own unique wastes of software development, but they are not explicitly targeted towards communication. However, some of the wastes from this literature are also valid in the context of communication (paper V). Since the concept of waste in software development is relatively less studied, its nature is somewhat elusive. On the other hand, existing literature on communication challenges is rather copious. According to the Merriam-Webster online dictionary, a challenge is defined as a "difficult task or problem: something that is hard to do[7]". Using this definition, a challenge in communication is something that makes it difficult. On the other hand waste does not by definition explicitly make communication challenging. It was found (paper V) that waste was something that did not in fact cause the project to fail, which supports the finding made by Ikonen (2010). However, if e.g.

[7] http://www.merriam-webster.com/dictionary/challenge

shared understanding is not achieved this can potentially have dramatic consequences for the project. Hence, the definition of waste in the context of communication, and software development in general, should be reformulated.

This study also indicated that *lack of trust between the partners is potentially the single most important factor threatening communication in distributed agile development*. Issues related to trust in distributed environments appear to be commonplace, as demonstrated e.g. by da Silva et al. (2010) in section 2.4 of this thesis. Further, Berger (2007) reported that trust issues can prevent stakeholders of an agile project from building and fostering a relationship enabling collaboration and cooperation, both of which are essential in agile development. It can be assumed that this defective relationship also affects communication between the participants.

From the perspective of communication in agile development, the lack of face-to-face communication and direct communication hinders building of trust between distributed partners (Conchúir et al. 2009, Therrien 2008, Lee & Yong 2010). In distributed environments face-to-face communication can be difficult to achieve and, hence, Therrien (2008) proposed regular visits between the distributed sites in order to build trust and Moe and Šmite (2008) proposed that teams with low trust should utilize frequent and predictable communication in order to maintain and improve the level of trust. Moe and Šmite (2008) argued based on existing work that trust is a multifaceted problem in GSD, since it is affected by factors related to geographical, organizational, temporal, cultural and political differences between the team members. From this perspective, the identified challenges of geographical, temporal and cultural distance and organizational factors in terms of bureaucracy can have a negative effect on trust and, hence, customer communication. Further, disparities between work practices, e.g. differences between traditional and agile methods (paper IV) can have a negative impact on trust Moe and Šmite (2008). Therefore, it appears that there is no single reason responsible for the lack of trust in distributed agile development projects.

This study revealed indications of lack of trust (papers III and IV). During one of the case projects reported in paper III, the customer was on-site with the remote team for the first two weeks of the project. After this, all the communication between the distributed teams was directed via the customer. However, the customer was not communicating with the remote team that would have required information, and direct communication between the distributed development teams was prohibited. It was suggested that the lack of trust between the distributed partners could have contributed to the lack of communication. The findings made in paper IV describe a similar situation. Although in this case the customer organization was actively communicating with the vendor unit, from their perspective the customer organization did not provide necessary information about requirements.

There were similar findings in the cases reported in papers III and IV. In paper IV, information was deliberately hidden from the vendor organization (i.e. they were not allowed to access the program code developed by the customer organization) due to its business sensitive nature, despite the fact that both organizations belonged to the same company. Similar information hiding in the form of

prohibiting direct communication between the teams was found in the case reported in paper III. Whereas the lack of trust was explicitly mentioned in the case reported in paper IV, it was only speculated in paper III. However, this similarity strengthens the case for lack of trust in this case. Further, existing literature provides insights to this problem. According to Dirks and Ferrin (2001), lack of trust can make working towards a single shared goal difficult and it is also probable that competitive motives will be given more attention than cooperation among the employees, and partners may even withdraw from collaboration due to feelings of insecurity (Bandow 2001). In addition, a low level of trust can also lead to reduced information exchange and feedback, as suggested in (Bandow 2001, Dirks & Ferrin 2001, Salas et al. 2005).

The findings of this study indicated that lack of trust is not limited to customer relationships where there is a separate customer and vendor organization, but also involves organizational branches that are in fact part of the same company. This suggests that lack of trust is a phenomenon that is not limited to any particular organizational context. Lack of trust is by no means a new finding in distributed development, e.g. (Conchúir et al. 2009, Therrien 2008, Lee & Yong 2010). However, Paasivaara et al. (2008b) found that Scrum practices improve trust in globally distributed efforts, since remote sites are able to conduct demanding tasks and more frequent communication improves monitoring progress. Further, it has been claimed that direct contact between the customer and the team establishes mutual trust (Sillitti et al. 2005). Hence, the findings of this study related to trust and the lack of it appear to contradict with existing literature. However, given the notion that the distributed environment as a problem domain is very complex (Komi-Sirviö & Tihinen 2005, Lee et al. 2006), this is not necessarily surprising. However, this work provides support for the hypothesis that lack of trust can be a key factor hindering or even preventing customer communication in distributed agile developments. Further, the claim made by Sillitti et al. (2005) should be regarded critically, since despite direct contact with the customer, trust was lacking (paper IV). Rather, it should be ensured that an environment is created that enables and fosters customer communication in distributed agile development (paper IV).

6.1.1 Media Richness Theory and Media Synchronicity Theory

This study used communication theories as lenses for gaining deeper understanding about communication in the case projects. Communication theories were explicitly used in the case projects reported in papers II, IV and V.

6.1.1.1 Media Richness Theory.

Media Richness Theory was used as a tool for analysing communication in papers II and IV. Paper II provided some evidence that less rich communication channels are not suitable for clarifying ambiguous issues. After the first two weeks when the customer was onsite during case 4, communication was conducted via email and

occasionally by telephone. This resulted in high numbers of defects. In addition, it was reported in paper II that face-to-face communication during requirements analysis (i.e. rich communication) created lively discussions between the customer and the developers. This helped to clarify requirements and supports the claim that rich communication is able to clarify ambiguous matters.

Paper IV was a practitioner-oriented publication and in this study Media Richness Theory was used as a lens to analyse communication. Several different media were used in communication between the distributed sites. However, these media were not solving the issue of communication being too high level to have been useful to the vendor site. Therefore, this paper could not show any supporting or contradicting evidence for MRT. The findings also indicate that communication needs to be meaningful (i.e. it should serve its intended purpose, in this case to provide sufficient information about the requirements) before communication theories can be applied in analysing and improving communication.

6.1.1.2 Media Synchronicity Theory

Media Synchronicity Theory was applied in the study reported in paper V. In this study, MST was able to explain the observed results. There were indications that understanding requirements would have necessitated more efficient conveyance in the form of more detailed requirements while storing them into the product backlog. However, this is counterintuitive to agile development, which suggests that requirements should be clarified when their implementation becomes timely. Furthermore, the use of various communication media helped to clarify unclear matters between the lead site in North America and the remote site in India. The issues stemmed from a lack of shared understanding considering the project, the need to explicitly document the requirements for the Indian site and issues related to a language barrier. These findings support the proposition of MST that no single communication medium is better than any other, but that different media should be applied on the basis of communication needs. Considering the communication media, this paper concluded that face-to-face communication is not necessarily the most efficient medium under certain conditions. In this project, one of the sites involved in the development effort was an external contractor who did not share similar domain knowledge with the other two sites. The case organization had to carefully explain to the contractor site the contents of the requirements they were allocated in very high detail. If this was not done, the features did not meet the expectations assigned to them. This suggests that in cases when there are great differences in domain and/or product knowledge, more detailed descriptions of requirements (for example) are a better option than face-to-face communication even in agile development projects.

This finding is in line with the suggestions of MST: if the communicating participants do not share a common understanding, a media with lower synchronicity is a better option to convey information. Hence, it can be proposed that MST could be used as an instrument for analysing and improving customer communication in distributed agile software development, and that communication strategies and

practices can be derived from it. It should however be noted that the use of MST in this study was limited to *conveyance* and *convergence* processes only.

6.2 Implications for practice

In this section, the key contribution of this study to practice is discussed.

The key contribution to practice is the toolbox depicted in Figure 8. This toolbox was first defined by the existing literature on challenges and solution proposals in customer communication and communication in general (Figure 7). The toolbox was complemented based on the findings of this study. This study identified the following challenge categories and solution proposals, which are now discussed against the background of existing literature.

A *Bureaucratic organization* challenge emerged from the case project reported in paper IV and refers to communication hindrances stemming from the bureaucratic nature of the customer organization. Earlier, Berger (2007) identified that a bureaucratic environment is challenging for the success of an agile development project. Bureaucracy is an organizational form that is based on strict rules, specified roles and positions (Hofstede 2003). Bureaucratic culture for its part is procedural, hierarchical, regulated, established, structured, cautious and power-oriented (Wallach 1983). There were several indications of bureaucratic characteristics prevalent in the customer organization. Cautiousness was identified as not openly sharing information with the vendor organization, and there were indications of culpability since there were fears of losing jobs if the project should fail, and of tightly defined responsibilities which prevented people from “crossing borders”. This latter problem was indicated in the form of deliberately preventing the developers from each organization from communicating directly with each other. This is counter-intuitive to agile methods that promote open and active communication between all the participants. Since customer communication was lacking and the vendor organization lacked knowledge of the domain, deliberate prohibition of communication resulting from bureaucratic environment posed severe threats to the success of the project.

In order to answer to the challenges created by bureaucratic organizational environment, a set of guidelines was proposed (paper IV). These guidelines can also be applied while dealing with communication challenges related to a challenge of *differences between traditional and agile development* (paper IV). The importance of the customer’s role is recognized in traditional methods and it is conducted using formal communication focusing on contractual matters on an as needed basis (Boehm & Turner 2003b, Nerur et al. 2005). By contrast, in agile methods the customers’ role is critical, customer interaction is driven by focused increments and communication is preferably conducted informally (Boehm & Turner 2003b, Nerur et al. 2005). The customer organization in the case project reported in paper IV was conducting their work in a traditional milestone-oriented manner, whereas the vendor organization had adopted an agile approach. One of the key characteristics of agile development is that communication focuses on steering the project

increment by increment. In this particular case, the customer organization did not want to be involved in “agile communication” and they did not participate in the planning and review meetings that took place at the vendor organization. Although these two organizations communicated very actively with each other, the customer organization did not clarify the requirements that were supposed to be implemented at the vendor site. This implies that when an agile team is working with a more traditionally oriented organization, it is possible that the agile team is not supported as it should be in terms of communication. Challenges related to adoption of agile methods have been discussed in the previous literature. For example Mahanti (2006) discussed issues at personal levels related to adoption of agile methods, whereas (Cohn & Ford 2003) approached adoption challenges from a managerial perspective. Further, Nerur et al. (2005) and Grossman et al. (2004) claimed that the most significant challenge when adopting agile methods lies in changing the organization as a whole in order to adopt an agile way of working. Since for many people involved in the case project (paper IV) this particular project was the first agile effort, it is reasonable to expect that agile methods were generally unfamiliar to the project participants. Hence, the abovementioned literature should provide insights into this challenge area and how it affected customer communication.

Another new challenge area is the *communication waste* discussed earlier in the previous section. These wastes should provide companies insights into what could be improved in customer communication in their individual contexts. This area takes a novel view to hindrances of communication in distributed agile development and the wastes can be mitigated and effects of mitigation actions evaluated by applying a systematic approach (paper V). This process is similar to the agile practice of Retrospectives, which focuses on continuous improvement of the development process. Similarly, communication should be actively analysed and improved. Although the process described in this case is based on analysing communication throughout individual process phases, it should be noted that the process does not explicitly take into account when and by whom this process should be conducted. Considering the principle of Self-organization which enables empowered people to independently make the solutions they see best, anyone in the project could be able to conduct the waste identification process.

Considering the earlier identified challenge of *lack of customer involvement*, new solution proposals were identified in this study. Lack of customer involvement was also identified as a communication challenge in this study and, hence, guidelines were applied to ensure that a distributed agile development project has an involved customer who provides necessary information to the development teams (paper IV). Further, paper I presented an empirically validated process for involving the customer in an agile development project. Moreover, a finding from a case reported in paper V provides means for integrating the customer to the development process: mandatory customer participation during the phases of the development project ensures efficient communication by systematically involving the customer in the process. In this particular case, the customer’s participation was mandatory during iteration planning and iteration review sessions and also during mid-iteration sessions that took place 2–3 times a week, in which the current in-

crement was tested by the customer. This enabled efficient customer communication in the project. Inarguably, this is very close to the on-site customer practice of XP (Beck 2000) and follows the *Real Customer Involvement* practice (Beck & Andres 2004) that has replaced on-site customer availability with less stringent requirements for participation. The approach taken in this case project clearly shows the importance of a dedicated customer for the rest of the development organization, since the scheduled meetings were postponed if the customer was not able to participate in them. It should, however, be noted that very often the customers can be too valuable for their employers or they can be too remote to participate in the development as expected (Jeffries et al. 2001). Furthermore, the customers can even be reluctant to participate in the process as actively as would be necessary while taking an agile development approach (Farell et al. 2002). Therefore, organizations aiming towards closer customer involvement should take into account the customer's ability and willingness to participate in a distributed agile project and identify those most essential phases in which customer involvement is essential and ensure that the customer is available when it is most important.

In this study, two distinct elements that impact customer communication emerged. These themes are the customer's involvement in the process and systematic analysis and improvement of customer communication. A customer's active participation in an agile process is one of the original key tenets of agile methods (Beck 2000). However, more recent literature has questioned the critical importance of customer involvement in agile development (Chow & Cao 2008, Stankovic et al. 2013). As discussed earlier in this work, Chow and Cao (2008) concluded that customer involvement plays an important role only in relation to the scope of the project, and in their study based on the work of Chow and Cao (2008), Stankovic et al. (2013) suggested that customer involvement is not important by any dimensions against which it was analysed. The findings of this study, however, are contradictory to those of Chow and Cao (2008) and Stankovic et al. (2013). The findings of this study clearly show that customer involvement is critical, since without involvement, customer communication can even be nonexistent, which in turn can jeopardise the whole project (papers III and IV). In the case reported in paper I the original customer was not able to participate in the process as necessary, which resulted in the appointment of a customer proxy consisting of two other customer representatives in order to maintain active customer communication. Further, the lack of customer involvement and decreasing communication correlate with the quality of the software being built in agile projects; the number of defects increases when the customer's involvement and communication decreases (paper II). Hence, involving the customer in the project is essential in agile projects, which was also shown in paper V. The claim of Chow and Cao (2008) considering the elusive and changing nature of the success factors in agile development can also be criticised. The individual studies in this work were conducted between 2004 and 2011, which gives a longitudinal perspective to this finding. Customer involvement was seen to be paramount in all the studies as a factor contributing to efficient communication. Therefore, given the emphasized

importance of customer communication in agile development and the role which customer involvement plays in it, in the light of this study customer involvement has always been a critical factor contributing to the success of an agile project.

Considering the analysis and improvement of customer communication, existing literature discussed in this study has shown that different communication media have their downsides. Similar findings were also made in this study considering the use of communication media. In the case reported in paper I, the customers requested daily emails that summarized what had been done during the day. The daily status emails were at first too technically oriented, and therefore more general level descriptions were requested by the customers. A similar format was used in the study reported in paper II. In this latter case, however, the daily reports were too abstract for the customer to provide feedback based on them. A finding considering the reports in paper I could provide some insights considering the abstract nature of the reports. In this study, the reports contained information only related to what was done and no questions were asked from the customers. The customers experienced that they did not have any reason to provide feedback. Despite this, the customers were not willing to use any other communication media during the iterations. Paper V identified challenges related to face-to-face communication, videoconferencing, email and other asynchronous communication media. In addition, it was suggested in paper III that communication practices can have their downsides, despite their potential for supporting customer communication. Although these findings are hardly new, together with existing literature they suggest that no single communication media or practice is a “silver bullet” that is able to solve all the issues in customer communication without drawbacks. However, it was suggested in paper V that the realities of software development (e.g. customers’ busy schedules) can dictate the use of communication media that are not necessarily the best for the given situation. Despite this, the means, practices and tools used in customer communication should be systematically evaluated throughout the project. Constant improvement is one of the key aspects of agile software development and this principle should also be applied in the context of customer communication. What also calls out for systematic evaluation is the claim made by McLuhan (1964) that the use of media is not static but, instead, the use should be applied to the situation in which the media are used. Similarly, Media Synchronicity Theory suggests that no single communication medium is inherently better than any other. Instead, the selection of appropriate media simultaneously or in succession creates the best possible communication results.

In addition, this toolbox provides a means to answer the critique that Petersen and Wohlin (2009) presented against agile methods related to communication overheads stemming from the high number of teams in large scale agile development projects. The toolbox enables both teams and managers (and customers) to foresee potential communication challenges in distributed agile development projects and also to respond to them accordingly. Hence, the toolbox supports communication in large agile projects.

7. Conclusions

This section summarizes the results of this thesis, discusses its limitations and suggests future research avenues. The results are presented by providing answers to the research questions presented in Section 1.1.

7.1 Answers to research questions

In the following, the answers to the research questions are provided. Each subquestion is addressed in separate sections based on the individual findings of the original research papers. Finally, the answer to the main research question of this thesis is provided.

7.1.1 Q1: Why is customer communication important in distributed agile software development?

The answer to this research question is as follows: this study has shown that customer communication is important since a lack of communication can cause issues related to software quality (paper II) and significant challenges even within small-scale distributed agile development projects (paper III). In the context of globally distributed agile projects in which both agile and traditional development methods are used, the combination of a customer who is not participating actively in the development, bureaucratic organizational environment, differences between traditional and agile methods and lack of trust can make the distributed agile development very challenging. The lack of trust in particular appears to be the single most important factor threatening communication in distributed agile environments. (Paper IV)

7.1.2 Q2: How is it possible to involve the customer in the development process in distributed agile development in order to ensure communication and feedback?

The answer to this research question is as follows: existing literature from distributed agile development (Layman et al. 2006) suggests the advantage of having a customer that must be able to make conclusive decisions on project's scope and

functionality, must have a vested interest in the project and needs to be readily available. A process aiming towards active customer involvement is presented in paper I. In addition, paper V proposes mandatory customer involvement. Based on these findings, it should be ensured that a committed customer is actively involved in distributed agile development project in order to ensure communication and feedback.

7.1.3 Q3: What are the means, practices and tools for improving customer communication in distributed agile development?

The answers to this research question are the challenge mitigation approaches for different challenges depicted in the toolbox (Figure 8). These approaches were derived from the existing literature and were complemented based on the results of this study.

These approaches identified from existing literature are as follows: use of synchronized communication media, use of communication media suitable for their intended purposes, application of communication policies, selective participation in meetings, minimized number of meetings, synchronized working hours, involvement of relevant people, visits, formal and detailed documentation, Scrum practices, frequent and predictable communication, communicate expectations early and establish rules for conflict handling, competence and language proficiency, a development approach that enables frequent communication, investments in suitable communication tools and infrastructure, communication theories and committed customers.

The approaches identified in this study are guidelines for ensuring communication (paper IV), a waste mitigation approach (paper V) and systematic customer integration (papers I and V).

7.1.4 RQ: How is it possible to improve customer communication in distributed agile software development?

Customer communication in distributed agile software development can be improved by taking into account the various challenges that can hinder communication. These are the challenge areas that are presented in the toolbox described in Section 5.6. These challenges provide an overview to potential communication challenges that an agile project may encounter. Subsequently, these challenges can be mitigated on the basis of the means, practices and tools described in the previous section. The challenges derived from the existing literature were geographical distance, temporal distance, cultural distance, lack of trust, communication tools and media selection, and lack of customer involvement. The challenges identified in this study are bureaucratic organization (paper IV), communication waste (paper V) and differences between traditional and agile approaches.

7.2 Trustworthiness and limitations of the study

As Creswell and Miller (2000) pointed out, addressing the validity of qualitative studies can be both challenging and confusing due to the wide array of terms associated with validity. However, the general consensus is that qualitative studies need to demonstrate credibility (Creswell & Miller 2000). In this section different ways of addressing validity are discussed, as well as the validity of this study as seen from the perspective of trustworthiness. In addition, some limitations of the study are discussed.

Yin (2003) approached validity from the perspectives of construct validity, internal validity, external validity and reliability. These criteria address validity as follows:

- *Construct validity* reflects to what extent the studied measures represent the intents of the research and what is being studied according to research questions. There is a threat to construct validity if the interviewees do not interpret the constructs discussed in the interviews in a same way as the researcher does.

- *Internal validity* is a criteria related to causal relationships. For example, if it is being studied whether a certain factor affects the factor being studied, there is a risk that a third factor also affects the one under study. There is a threat to internal validity if the researcher is not aware of this factor and/or does not know to what extent this third factor affects the investigated one.

- *External validity* is an aspect related to generalizability of the results and to the extent of interest in the results among people outside the investigated case. In case studies, the intention is to extend the results to similar cases in which the findings would be relevant.

- *Reliability* considers the extent to which the data and the analysis are dependent on the researcher. If another researcher were to conduct the same study later, the findings should be the same. For example, if it is unclear how the data has been coded and if interview questions and/or questionnaires are unclear, there is a threat to the reliability of the study.

In addition to these criteria, Guba (1981) proposed internal validity, external validity, reliability and objectivity as important criteria for addressing validity. However, as Stol et al. (2014) pointed out, these criteria are more suitable for evaluating quantitative data and, in turn, qualitative research should be evaluated in terms of *trustworthiness* using *credibility¸ transferability, dependability* and *confirmability* as the criteria against which trustworthiness is assessed. In order to support this, Cruzes and Dybå (2011) proposed addressing trustworthiness using these criteria

in the context of software engineering. These criteria are described as follows based on Stol et al. (2014) and Cruzes and Dybå (2011):

- *Credibility* is related to the confidence and plausibility of the findings. Therefore, the focus of the research and the confidence in how well the data and the analysis processes address the intended focus are important.

- *Transferability* addresses the generalizability of the results, i.e. to what extent the findings can be applied in other settings.

- *Dependability* refers to the stability of data, i.e. the extent to which findings are reliable, the degree to which the data changes over the course of time and as a result of alterations in the researcher's decisions during the data analysis process. It should be possible to track and explain any variance in findings.

- *Confirmability* is related to coding and sorting of extracted data and it also considers whether other researchers would agree with the applied coding and sorting approach. Confirmability also concerns bias from the perspectives of researcher and participants.

Since four out of the five papers included in this work are based on qualitative data and the context of the study is software engineering, the trustworthiness of the study is evaluated as suggested by Cruzes and Dybå (2011).

In order to improve *credibility* the studies, collected and analysed data and the findings were actively discussed with researchers involved in the studies. This peer debriefing (Creswell & Miller 2000) provided valuable feedback that could be utilized during the individual studies. This approach is also called *venting* through which results and interpretations are discussed with colleagues to prevent the problem of *multiple realities* (Kaplan & Duchon 1988, Goetz & LeCompte 1984). According to Stol et al. (2014), triangulation is a technique seeking convergence among different and multiple information sources. These sources can be different investigators, different research methods and different types of data. In the studies of this thesis, the data was collected from several data sources during all but one study. Furthermore, during the study reported in paper V, the data was member checked with the participants, which according to Lincoln and Guba (1985) is the most crucial aspect to be taken into account when establishing credibility. In paper I member checking was conducted iteratively during the research in the project's iteration retrospectives. In the study reported in paper III, member checking was conducted to a certain extent in the form of a single interview conducted after the project had ended. In other studies, member checking with the participants was not possible. This can be seen as a threat to credibility.

Lee and Baskerville (2003) claimed that many researchers working with both quantitative and qualitative data have restricted themselves only to statistical,

sampling-based generalizability. Since statistical generalization cannot be derived from this study, *transferability* is addressed from the perspectives of four alternative types of generalization as suggested by Walsham (1995). These types are *development of concepts, generation of theory, drawing of specific implications* and *contribution of rich insights.*

Considering *development of concepts*, paper I presented a process for integrating the customer to an agile development process in order to ensure active customer communication. This process can be considered by organizations as a means for involving the customer in the development process. Whereas Orlikowski and Baroudi (1991) stated that interpretive studies do not seek generalizability, the types of generalization discussed by Walsham (1995) are explicitly discussed in the context of interpretive research. Taking this into account, it can be claimed that the waste identification process is another concept developed during this study.

The aim of this study was not to explicitly create a theory, or some propositions in the form of *theory fragments*, i.e. a partial theory that is not yet completely developed (Stol & Fitzgerald 2013). In this work, it was suggested that the lack of trust in the context of distributed agile software development might be the key obstacle for customer communication. This can be seen as an initial tentative theory, which should however be studied further.

Considering the *drawing of specific implications,* this study provided several implications for both research and practice. Finally, since the main instrument of data collection in all but one study was interviews, this contributes to *contribution of rich insights*. The individual studies have aimed to provide as rich a description of the case projects and case data as possible. However, similarly to the limitation presented in Stol et al. (2014), it is challenging to provide all the details of individual case studies in the research papers due to manuscript length limitations. This affects negatively the "feeling of experience" discussed by Creswell and Miller (2000). According to these authors, "thick" descriptions (i.e. rich insights) create statements, through which the readers could experience, or have experienced, the events that the study describes.

In order to address *dependability* several data sources were used to provide the results in all but one study. The material collected from each individual study forms an audit trail which helps other investigators "to follow the cognitive development of a project as it developed" (Morse 1994, p. 24). However, there are limits to dependability. In the cases reported in papers II, III and IV, it was not possible to interview the customers, which is a threat to dependability. In addition, during the study reported in paper II, defect data recording was conducted accurately by the developers only in one of the individual cases. Hence, the defect data from the three other cases was able to show only trends in defect amounts. This is a threat to dependability.

The *Confirmability* aspect can be addressed with similar techniques as *credibility*. As discussed in the context of credibility, peer debriefing was applied between the researchers and the data was triangulated. Further, member checking was

applied in some of the cases when it was possible. Hence, this is a threat for the confirmability of the study.

Considering the threats to individual aspects and, hence, the trustworthiness of the study, the concept of *convenience sampling* should be noted. According to Lethbridge et al. (2005), most studies in the field of software engineering need to rely on participants who are available and willing to participate in the study. This will however lead to bias, and therefore studies need to acknowledge this. Slightly biased data is still seen as much more useful than a lack of data altogether Lethbridge et al. (2005), but it should be noted that the results reported in individual research papers and, hence, the results of this study can be biased due to limited availability of participants during the study, considering the inability to interview the customers during the cases reported in papers II, III and IV.

This thesis has other limitations as well. In the studies reported in papers I, II and III the developers involved in the case projects were mainly advanced MSc level students from the field of software engineering. However, several authors have discussed the use of students in software development research. For example, Höst et al. (2000) came out in favour of using students when experimenting in software engineering. Considering the papers I, II and III, the Mobile-D development approach was applied in the case projects from which the data was collected. The case projects involved several researchers with individual research interests and at the general level, these studies aimed towards further development and validation of Mobile-D, which is a highly experimental development approach. Since the nature of the case projects as a whole was experimental, the use of student subjects is acceptable considering the claim of Höst et al. (2000). For example Kitchenham et al. (2002) did not propose to neglect the empirical data collected from projects in which students are involved.

7.3 Future research opportunities

This thesis focused on customer communication in distributed agile software development at the project level. It could be worthwhile to study communication and verify the results of this thesis in a wider context, involving the whole organization. In addition, it could be worthy of study to assess what lean software development practices or elements from novel development approaches, such as “Leagile” discussed in Section 2.5 or other interesting methods, for example Scrumban (Ladas 2009), could possibly provide for communication. Other interesting research avenues also exist. Both Media Richness Theory and Media Synchronicity Theory were used as the communication theories in the original research papers. By constructing a theoretical model based on these theories for selecting the most efficient communication solutions in given situations it could be possible to provide companies with more insights into customer communication in their work. This contribution would also benefit the scientific community.

Considering the discussion of the vague nature of waste in Section 6.1, the nature of waste should be studied in depth. Section 6.1 also called out for a more

refined definition of this concept and, hence, the use of a preliminary definition of waste being *something that consumes resources but does not add value and can potentially cause problems if not mitigated* could be used as a starting point for studying this concept further. Section 6.1 emphasized lack of trust as a key factor affecting customer communication. Given this claim, it would be important to study whether or not lack of trust is the key obstacle hindering and even preventing customer communication in agile software development. Finally, the presented toolbox should be empirically evaluated in distributed agile development projects.

References

Abbas, N., Gravell, A. M. & Wills, G. B. (2008). Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From? In: the proceedings of 9th International Conference in Agile Processes in Software Engineering and Extreme Programming (XP2008). June 10–14, 2008. Limerick, Ireland. pp. 94–103.

Abrahamsson, P., Conboy, K., & Wang, X. (2009). Lots done, more to do: the current state of agile systems development research. European Journal of Information Systems, 18(4), pp.281–284.

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P. & Salo, O. (2004). Mobile-D: An Agile Approach for Mobile Application Development. In: The proceedings of the 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA.04). October 24–28, 2004. Vancouver, British Columbia, Canada. pp. 174–175.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. VTT Publications 478. VTT, Espoo. www.vtt.fi/inf/pdf/publications/2002/P478.pdf

Ågerfalk, P. & Fitzgerald, B. (2006). Flexible and Distributed Software Processes: Old Petunias In New Bowls? Communications of the ACM, 49(10), pp. 10–27.

Ågerfalk, P.J. (2004). Investigating actability dimensions: a language/action perspective on criteria for information systems evaluation. Interacting with Computers,16(5), pp. 957–988.

Ågerfalk, P., Fitzgerald, B. & Slaughter, S. (2009). Flexible and distributed information systems development: State of the art and research challenges. Information systems research, 20(3), pp. 317–328.

Agile Manifesto (2001). [online] Available at: http://agilemanifesto.org/

Avison, D.E. & Fitzgerald, G. (2003). Where now for development methodologies? Communications of the ACM, 46(1), pp. 78–82.

Bakalova, Z. & Daneva, M. (2011). A comparative case study on clients participation in a' traditional' and in an Agile software company. In: Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement (PROFES 2011). June 20–22, 2011. Torre Canne, Italy. pp. 74–80.

Balzer, R., Cheatham Jr, T.E. & Green, C. (1983). Software Technology in the 1990's: Using a New Paradigm. Computer, 16(11), pp. 39–45.

Bandow, D. (2001). Time to create sound teamwork. Journal for Quality and Participation, 24(2), pp.41.

Bannerman, P. L., Hossain, E. & Jeffery, R. (2012). Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage? In: Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS 2012). January 4–7, 2012. Maui, HI, USA. pp. 5309–5318.

Baskerville, R. L. (1999). Investigating Information Systems with Action Research. Communication of the Association for Information Systems, 2(19), October 1999.

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

Beck, K. (2000). Extreme Programming Explained: Embrace change. Addison-Wesley, Upper Saddle River, New Jersey. 190 p.

Beck, K. (1999). Embracing Change with Extreme Programming. IEEE Computer, 32(10), pp. 70–77.

Beck, K. & Andres, C. (2004). Extreme Programming Explained: Embrace Change. Second Edition. Addison-Wesley, Upper Saddle River, NJ, USA. 189 p.

Benington, H.D. (1983). Production of Large Computer Programs. Annals of the History of Computing, 5(4), pp. 350–361.

Berger, H. (2007). Agile development in a bureaucratic arena – a case study experience. International Journal of Information Management, 27(6), pp. 386–396.

Boehm, B. (2003). Value-Based Software Engineering. ACM SIGSOFT Software Engineering Notes, 28(2), pp. 1–12.

Boehm, B. (2002). Get Ready For Agile Methods, With Care. IEEE Computer, 35(1), pp. 64–69.

Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. Computer, 21(5), pp. 61–72.

Boehm, B. & Turner, R. (2003a). People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods. Crosstalk – The Journal of Defense Software Engineering, (Dec 2003), 16(12), pp. 4–8.

Boehm, B.W. & Turner, R. (2003b). Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Professional. 266 p.

Boland, D. & Fitzgerald, B. (2004). Transitioning From a Co-located to a Globally-distributed Software Development Team: A Case Study at Analog Devices Inc. In: the proceedings of the 26th International Conference on Software Engineering (ICSE 2004). The 3rd International Workshop on Global Software Development. May 28, 2004. Edinburgh, UK. pp. 4–7.

Börjesson, A. & Mathiassen, L. (2005). Improving software organizations: agility challenges and implications. Information Technology & People, 18(4), pp. 359–382.

Bostrom, R. B. & Thomas, B. D. (1983). Achieving excellence in communications: A key to developing complete, accurate and shared information requirements. In: The Proceedings of the Twentieth Annual Computer Personnel on Research Conference (SIGCPR '83). November 17–18, 1983. Charlottesville, VA, USA. pp. 1–13.

Bowers, J., May, J., Melander, E., Baarman, M. & Ayoob, A. (2002). Tailoring XP for large system mission critical software development. In: the Proceedings of the Second XP Universe and First Agile Universe Conference. August 4–7, 2002. Chicago, IL, USA. pp. 100–111.

Braithwaite, K. & Joyce, T. (2005). XP expanded: Distributed extreme programming. In: the proceedings of the 6th International Conference in Extreme Programming and Agile Processes in Software Engineering (XP2005). June 18–23, 2005. Sheffield, UK. pp. 180–188.

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative research in psychology, 3(2), pp. 77–101.

Bruegge, B., Dutoit, A. H. & Wolf, T. (2006). Sysiphus: Enabling informal collaboration in global software development. In: the proceedings of International Conference on Global Software Engineering (ICGSE '06). October 16–19, 2006. Florianopolis, Brazil. pp. 139–148.

Cao, L., Mohan, K., Xu, P. & Ramesh, B. (2004). How extreme does extreme programming have to be? Adapting XP practices to large-scale projects. In: the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS 2004). January 5–8, 2004, Big Island, HI, USA. pp. 1–10.

Chow, T. & Cao, D. (2008). A survey study of critical success factors in agile software projects. Journal of Systems and Software, 81(6), pp. 961–971.

Cockburn, A. (2002). Agile Software Development. Addison-Wesley, Indianapolis. 278 p.

Cohn, M. & Ford, D. (2003). Introducing an Agile process to an Organization. IEEE Computer, 36(6), pp. 74–78.

Conboy, K. (2009). Agility from first principles: reconstructing the concept of agility in information systems development. Information Systems Research, 20(3), pp. 329–354.

Conboy, K. & Fitzgerald, B. (2007). Agile Drivers, Capabilities, and Value: An Over-Arching Assessment Framework for Systems Development. In: *Agile Information Systems: Conceptualization, Construction, and Management*. DeSouza, K.C. (ed.), Elsevier, Burlington, MA, USA. pp. 207–222.

Conboy, K. & Fitzgerald, B. (2004). Toward a conceptual framework of agile methods: a study of agility in different disciplines. In: Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research (WISER '04) co-located with SIGSOFT 2004/FSE 12 Conference. November 5, 2004. Newport Beach, CA, USA. pp. 37–44.

Conchúir, E.Ó., Ågerfalk, P.J., Olsson, H.H. & Fitzgerald, B. (2009). Global software development: where are the benefits? Communications of the ACM, 52(8), pp. 127–131.

Counsell, S., Phalp, K., Mendes, E. & Geddes, S. (2005). What Formal Models Cannot Show Us: People Issues During The Prototyping Process. In: the proceedings of the 6th International Conference on Product Focused Software Process Improvement (PROFES 2005). June 13–18, 2005. Oulu, Finland. pp. 3–15.

Creswell, J.W. & Miller, D.L. (2000). Determining validity in qualitative inquiry. Theory into practice, 39(3), pp. 124–130.

Cruzes, D.S. & Dybå, T. (2011). Recommended steps for thematic synthesis in software engineering. In: the proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement (ESEM 2011) September 22–23, 2011. Banff, Alberta, Canada. pp. 275–284.

Cunningham, J.B. (1997). Case study principles for different types of cases. Quality and quantity, 31(4), pp. 401–423.

Curtis, B., Krasner, H. & Iscoe, N. (1988). A Field Study of the Software Design Process for Large Systems. Communications of the ACM, 31(11), pp. 1268–1287.

Da Silva, F.Q.B., Costa, C., França, A.C.C. & Prikladinicki, R. (2010). Challenges and Solutions in Distributed Software Development Project Management: A Systematic Literature Review. In: the proceedings of the 5th International Conference on Global Software Engineering (ICGSE 2010). August 23–26, 2010. Princeton, NJ, USA. pp. 87–96.

Daft, R.L., Lengel, R. & Trevino, L.K. (1987). Message Equivocality, Media Selection, and Manager Performance: Implications for Information Support Systems. MIS Quarterly, 11(3), pp. 355–366.

Daft, R.L. & Lengel, R.J. (1986). Organizational Information Requirements, Media Richness and Structural Design. Management Science, 32(5), pp. 554–571.

Daft, R.L. & Weick, K. (1984). Toward a Model of Organizations as Interpretation Systems. Academy of Management Review, 9(2), pp. 284–295.

Damian, D. & Moitra, D. (2006). Guest Editors' Introduction: Global Software Development: How Far Have We Come? IEEE Software, 23(5), pp. 17–19.

Damian, D. (2002). Workshop on global software development. SIGSOFT Software Engineering Notes, 27(5).

Danait, A. (2005). Agile Offshore Techniques-A Case Study. In: the proceedings of the Agile Development Conference (ADC'05). July 24–29. Denver, CO, USA. pp. 214–217.

Davison, R., Martinsons, M.G. & Kock, N. (2004). Principles of canonical action research. Information Systems Journal, 14(1), pp. 65–86.

De Meyer, A., Loch, C.H. & Pich, M.T. (2002). From variation to chaos. MIT Sloan Management Review, 43, pp. 60–67.

Del Nuevo, E., Piattini, M. & Pino, F.J. (2011). Scrum-based Methodology for Distributed Software Development. In: the proceedings of the 6th International Conference of Global Software Engineering (ICGSE 2011). August 15–18. 2011. Helsinki, Finland. pp. 66–74.

Dennis, A.R., Fuller, R.M. & Valacich, J.S. (2008). Media, Tasks, and Communication Processes: A Theory of Media Synchronicity. MIS quarterly, 32(3), pp. 575–600.

Dennis, A.R. & Valacich, J.S. (1999). Rethinking Media Richness: Towards a Theory of Media Synchronicity. In: the proceedings of the 32nd Annual Hawaii International Conference on System Sciences (HICSS 1999). January 5–8, 1999. Maui, HI, USA. pp. 1017–1027.

Dijkstra, E.W. (1968). Letters to the editor: go to statement considered harmful. Communications of the ACM, 11(3), pp. 147–148.

Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N.B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), pp. 1213–1221.

Dirks, K.T. & Ferrin, D.L. (2001). The role of trust in organizational settings. Organization Science, 12(4), pp. 450–467.

Drummond, B.S. & Francis, J. (2008). Yahoo! Distributed agile: notes from the world over. In: Proceedings of Agile 2008. August 4–8, 2008, Toronto, ON, Canada, pp. 315–321.

Dybå, T. & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50(9–10), pp. 833–859.

Easterbrook, S., Singer, J., Storey, M. & Damian, D. (2008). Selecting empirical methods for software engineering research. In: *Guide to advanced empirical software engineering*. F. Shull, J. Singer & D.I. Sjoberg (eds.) Springer, pp. 285–311.

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

Edstrom, A. (1997). User Influence and the Success of MIS Projects: A Contingency Approach. Human Relations, 30(7), pp. 580–607.

Eisenhardt, K.M. (1989). Building Theories From Case Study Research. Academy of management review, 14(4), pp. 532–550.

Erickson, J., Lyytinen, K. & Siau, K. (2005). Agile modeling, Agile Software Development, and Extreme Programming: the State of Research. Journal of Database Management, 16(4), pp. 88–100.

Estler, H.C., Nordio, M., Furia, C.A., Meyer, B. & Schneider, J. (2012). Agile vs. Structured Distributed Software Development: A Case Study. In: the proceedings of the 7th International Conference on Global Software Engineering (ICGSE 2012). August 27–30, 2012. Porto Alegre, Brazil. pp. 11–20.

Farell, C., Narang, R., Kapitan, S. & Webber, H. (2002). Towards an Effective Onsite Customer Practice. In: the proceedings of the 3rd International Conference on XP and Agile Processes in Software Engineering (XP2002). May 26–29, 2002. Alghero, Sardinia, Italy. pp. 52–55.

Fish, R. S., Kraut, R. E., Root, R. W. & Rice, R. E. (1992). Evaluating video as a technology for informal communication. In: the Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 1992). May 3–7, 1992. Monterey, CA, USA. pp. 37–48.

Fitzgerald, B., Hartnett, G. & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. European Journal of Information Systems, 15(2), pp. 200–213.

Fitzgerald, B., Stol, K.-J., O'Sullivan, R. & O'Brien, D. (2013). Scaling Agile Methods to Regulated Environments: An Industrial Case Study. In: the Proceedings of 35th International Conference on Software Engineering (ICSE 2013). May 18–26, 2013. San Francisco, CA, USA. pp. 863–872.

Gilb, T. (1981). Evolutionary development. ACM SIGSOFT Software Engineering Notes, 6(2), pp. 17.

Gladden, G.R. (1982). Stop the life-cycle, I want to get off. ACM SIGSOFT Software Engineering Notes, 7(2), pp. 35–39.

Glass, R.L. (2006). The Standish report: does it really describe a software crisis? Communications of the ACM, 49(8), pp. 15–16.

Goetz, J.P. & LeCompte, M.D. (1984). Ethnography and qualitative design in educational research. Academic Press, Orlando, FL, USA.

Goldhaber, G. (1993). Organisational Communication. Sixth Edition. Brown & Benchmark Publishers.

Gorton, I. & Motwani, S. 1996. Issues in co-operative software engineering using globally distributed teams. Information and Software Technology, 38(10), pp. 647–655.

Grisham, P. S. & Perry, D. E. (2005). Customer relationships and extreme programming. ACM SIGSOFT Software Engineering Notes, 30. pp. 1–6.

Grossman, F., Bergin, J., Leip, D., Merritt, S. & Gotel, O. (2004). One XP experience: Introducing Agile (XP) Software Development into a Culture that is Willing but not Ready. In: proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research (CASCON 2004). October 5–7, 2004. Markham, Ontario, Canada. pp. 242–254.

Guba, E.G. (1981). Criteria for assessing the trustworthiness of naturalistic inquiries. Educational Technology Research and Development, 29(2), pp. 75–91.

Hanssen, G.K. & Fægri, T.E. (2006). Agile Customer Engagement: a Longitudinal Qualitative Case Study. In: the proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering (ISESE´06). September 21–22, 2006. Rio de Janeiro, Brazil. pp.164–173.

Hanssen, G.K., Šmite, D. & Moe, N.B. (2011). Signs of agile trends in global software engineering research: A tertiary study. In: 6th IEEE International Conference on Global Software Engineering Workshop (ICGSEW 2011). August 15–18, 2011. Helsinki, Finland. pp. 17–23.

Hansson, C., Dittrich, Y., Gustafsson, B. & Zarnak, S. (2006). How agile are industrial software development practices? The Journal of Systems & Software, 79(9), pp. 1295–1311.

Herbsleb, J. & Grinter, R. (1999). Splitting the Organization and Integrating the Code: Conway's Law Revisited. In: the proceedings of the 21st international conference on Software engineering (ICSE´99), May 16–22, 1999. Los Angeles, CA. USA. pp. 85–95.

Herbsleb, J. & Moitra, D. (2001). Global software development. IEEE Software, 18(2), pp. 16–20.

Highsmith, J. (2000). Adaptive Software Development: A Collaborative Approach To Managing Complex Systems. Dorset House Publishing, NY, USA.

Highsmith, J. & Cockburn, A. (2001). Agile Software Development: The Business of Innovation. Computer, 34(1), pp. 120–122.

Hofstede, G. (2003). Cultures and Organizations-Software of the Mind: Intercultural Cooperation and its Importance for Survival. Profile Books, London.

Holmström, H., Conchuir, E.O., Agerfalk, P.J. & Fitzgerald, B. (2006a). Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In: the proceedings of International Conference on Global Software Engineering (ICGSE ’06). October 16–19, 2006. Florianopolis, Brazil. pp. 87–95.

Holmström, H., Fitzgerald, B., Ågerfalk, P.J. & Conchuir, E.O. (2006b). Agile practices reduce distance in global software development. *Information Systems Development,* 23(3), pp. 7–18.

Hossain, E., Babar, M.A. & Paik, H. (2009). Using Scrum in Global Software Development: A Systematic Literature Review. In: the proceedings of the 4th International IEEE Conference on Global Software Engineering (ICGSE 2009). July 13-16, 2009. Limerick, Ireland. pp. 175–184.

Höst, M., Regnell, B. & Wohlin, C. (2000). Using Students as Subjects – A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. Empirical Software Engineering, 5(3), pp. 201–214.

Humphrey, W.S. (1995). A Discipline for Software Engineering. Addison-Wesley Longman Publishing Co., Boston, MA, USA.

Ikonen, M. (2010). Leadership in Kanban Software Development Projects: A Quasi-controlled Experiment. In: the proceedings of the International Conference on Lean Enterprise Software and Systems (LESS2010). October 17–20, 2010. Helsinki, Finland. pp. 85–98.

Ikonen, M., Kettunen, P., Oza, N. & Abrahamsson, P. (2010). Exploring the sources of waste in Kanban software development projects. In: the proceedings of the 36th Euromicro Conference on Software Engineering and Advanced Applications (EUROMICRO 2010). September 1–3, Lille, France. pp. 376–381.

Jalali, S. & Wohlin, C. (2012). Global software engineering and agile practices: a systematic review. Journal of Software: Evolution and Process, 24(6), pp. 643–659.

Jeffries, R., Anderson, A. & Hendrickson, C. (2001). Extreme Programming Installed. Addison-Wesley, Upper Saddle River, New Jersey, USA.

Jiang, L. & Eberlein, A. (2009). An analysis of the history of classical software development and agile development. In: the proceedings of 2009 IEEE International Conference on Systems, Man and Cybernetics (SMC 2009). October 11–14, 2009. San Antonio, TX, USA. pp. 3733–3738.

Jiménez, M., Piattini, M. & Vizcaíno, A. (2009). Challenges and improvements in distributed software development: a systematic review. Advances in Software Engineering, 2009, pp. 3–17.

Jokela, T. & Abrahamsson, P. (2004). Usability Assessment of an Extreme Programming Project: Close Co-operation with the Customer Does Not Equal to Good Usability. In: Proceedings of the 5th International Conference on Product Focused Software Development and Process Improvement (PROFES 2004). April 5–8, 2004. Kansai Science City, Japan. pp. 393–407.

Järvenpää, S.L., Shaw, T.R. & Staples, D.S. (2004). Towards contextualized theories of trust: the role of trust in global virtual teams. Information Systems Research, 15(3), pp. 250–267.

Kähkönen, T. (2004). Agile methods for large organizations-building communities of practice. In: the proceedings of Agile Development Conference 2004 (ADC 2004). June 22–26, 2004. Salt Lake City, UT, USA. pp. 2–10.

Kajko-Mattsson, M., Azizyan, G. & Magarian, M.K. (2010). Classes of Distributed Agile Development Problems. In: the proceedings of Agile 2010. August 9–13, 2010. Orlando, FL, USA. pp. 51–58.

Kaplan, B. & Duchon, D. (1988). Combining qualitative and quantitative methods in information systems research: a case study. Mis Quarterly, 12(4), pp. 571–586.

Keil, M. & Carmel, E. (1995). Customer-Developer Links in Software Development. Communications of the ACM, 38(5), pp. 33–44.

Kettunen, P. (2009). Agile Software Development in Large-Scale New Product Development Organization: Team-Level Perspective. Doctoral Disseration. Helsinki University of Technology, Espoo.

Kettunen, P. & Laanti, M. (2008). Combining agile software projects and large-scale organizational agility. Software Process Improvement and Practice, 13(2), pp. 183–193.

Kircher, M., Jain, P., Corsaro, A. & Levine, D. (2001). Distributed eXtreme Programming. In: the proceedings of XP2001. May 21–23, 2001. Villasimius, Sardinia, Italy, pp. 66–72.

Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K. & Rosenberg, J. (2002). Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering, 28(8), pp. 721–734.

Klein, H.K. & Myers, M.D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. MIS quarterly, 23(1), pp. 67–93.

Koch, A.S. (2005). Agile Software Development: Evaluating the Methods for Your Organization. Artech House Publishers, USA.

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

Koskela, J. & Abrahamsson, P. (2004). On-Site Customer in an XP Project: Empirical Results from a Case Study. In: the proceedings of the 11th European Conference on Software Process Improvement (EuroSPI 2004). November 10–12, 2004. Trondheim, Norway. pp. 1–11.

Kraut, R., Egido, C. & Galegher, J. (1988). Patterns of Contact and Communication in Scientific Research Collaboration. In: the proceedings of the 1988 ACM conference on Computer-Supported Cooperative Work (CSCW 1988). September 26–28, 1988. Portland, Oregon, USA. pp. 1–12.

Kraut, R.E. & Streeter, L.A. (1995). Coordination in Software Development. Communications of the ACM, 38(3), pp. 69–81.

Kruchten, P. (2004). The rational unified process: an introduction. Addison-Wesley Professional.

Kussmaul, C., Jack, R. & Sponsler, B. (2004). Outsourcing and Offshoring with Agility: A Case Study. In: the proceedings of the 4th Conference on Extreme Programming and Agile Methods (XP/Agile Universe 2004). August 15–18, 2004. Calgary, Alberta, Canada. pp. 147–154.

Laanti, M., Similä, J. & Abrahamsson, P. (2013). Definitions of Agile Software Development and Agility. In: the proceedings of the 20th European Conference on Software Process Improvement (EuroSPI 2003). June 25–27, 2013. Dundalk, Ireland. pp. 247–258.

Laanti, M., Salo, O. & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. Information and Software Technology, 53(3), pp. 276–290.

Ladas, C. (2009). Scrumban-essays on kanban systems for lean software development. Modus Cooperandi Press.

Larman, C. & Basili, V.R. (2003). Iterative and Incremental Development: A brief history. IEEE Computer, 36(6), pp. 47–56.

Lau, F. (1999). Toward a framework for action research in information systems studies. Information Technology & People, 12(2), pp. 148–176.

Layman, L., Williams, L., Damian, D. & Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. Information and Software Technology, 48(9), pp. 781–794.

Lee, A.S. & Baskerville, R.L. (2003). Generalizing generalizability in information systems research. Information systems research, 14(3), pp. 221–243.

Lee, G., DeLone, W. & Espinosa, J.A. (2006). Ambidextrous coping strategies in globally distributed software development projects. Communications of the ACM, 49(10), pp. 35–40.

Lee, S. & Yong, H.S. (2010). Distributed agile: project management in a global environment. Empirical Software Engineering, 15(2), pp. 204–217.

Lengel, R.H. & Daft, R.L. (1988). The Selection of Communication Media as an Executive Skill. Academy of Management Executive, 2(3), pp. 225–232.

Leon, G. (1995). On the diffusion of software technologies: technological frameworks and adoption profiles. In: the proceedings of the first IFIP WG 8.6 working conference on the diffusion and adoption of information technology, Oslo, Norway, October 1995, Oslo, Norway. pp. 97–116.

Lethbridge, T.C., Sim, S.E. & Singer, J. (2005). Studying software engineers: Data collection techniques for software field studies. Empirical Software Engineering, 10(3), pp. 311–341.

Lincoln, Y.S. & Guba, E.G. (1985). Naturalistic inquiry. Sage, Newbury Park, CA, USA.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J. & Kähkönen, T. (2004). Agile Software Development in Large Organizations. Computer, 37(12), pp. 26–34.

MacGregor, E., Hsieh, Y. & Kruchten, P. (2005). The impact of intercultural factors on global software development. In: the proceedings of the Canadian Conference on Electrical and Computer Engineering. May 1–4, 2005. Saskatoon, Sask. Canada. pp. 920–926.

Mahanti, A. (2006). Challenges in Enterprise Adoption of Agile Methods – A Survey. Journal of Computing and Information Technology, 14(3), pp. 197–206.

Malone, T.W. & Crowston, K. (1994). The Interdisciplinary Study of Coordination. ACM Computing Surveys (CSUR), 26(1), pp. 87–119.

Mandić, V., Oivo, M., Rodríguez, P., Kuvaja, P., Kaikkonen, H. & Turhan, B. (2010). What Is Flowing in Lean Software Development? In: the proceedings of the International Conference on Lean Enterprise Software and Systems (LESS2010). October 17–20, 2010. Helsinki, Finland. pp. 72–84.

Martin, A., Biddle, R. & Noble, J. (2004). The XP customer role in practice: three studies. In: the proceedings of Agile Development Conference 2004 (ADC 2004). June 22–26, Salt Lake City, UT, USA. pp. 42–54.

Maurer, F. (2002). Supporting distributed extreme programming. In: the proceedings of the 2nd XP Universe and First Agile Universe Conference (XP/Agile Universe 2002). August 4–7, 2002, Chicago, IL, USA. pp. 13–22.

McCracken, D.D. & Jackson, M.A. (1982). Life cycle concept considered harmful. ACM SIGSOFT Software Engineering Notes, 7(2), pp. 29–32.

McLuhan, M. (1964). Understanding Media: The Extensions of Man. McGraw Hill, New York.

Melnik, G. & Maurer, F. (2004). Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing. In: the proceedings of Agile Development Conference 2004 (ADC 2004). June 22–26, Salt Lake City, UT, USA. pp. 21–31.

Menand, L. (1997). Pragmatism: A reader. Vintage Press, New York.

Merisalo-Rantanen, H., Tuunanen, T. & Rossi, M. (2005). Is Extreme Programming Just Old Wine in New Bottles: A Comparison of Two Cases. Journal of Database Management (JDM), 16(4), pp. 41–61.

Miles, M.B. & Huberman, A.M. (1994). Qualitative Data Analysis: An Expanded Sourcebook. 2nd edn. SAGE Publications Inc., Thousand Oaks, California, USA.

Moe, N.B. & Šmite, D. (2008). Understanding a lack of trust in global software teams: A multiple-case study. Software Process Improvement and Practice, 13, 2008, pp. 217–231.

Moenaert, R.K. & Caeldries, F. (1996). Architectural redesign, interpersonal communication, and learning in R&D. Journal of Product Innovation Management, 13(4), pp. 296–310.

Morse, J. M. (1994). Emerging from the Data: The Cognitive Processes of Analysis in Qualitative Inquiry. In J. M. Morse (ed.) *Critical Issues in Qualitative Research Methods*. Thousand Oaks, CA: Sage. pp. 23–43.

Murru, O., Deias, R. & Mugheddue, G. (2003). Assessing XP at a European Internet company. IEEE Software, 20(3), pp. 37–43.

Nerur, S., Mahapatra, R.K. & Mangalaraj, G. (2005). Challenges of Migrating to Agile Methodologies. Communications of the ACM, 48(5), pp. 72–78.

Niederman, F., Kundu, S. & Salas, S. (2006). IT software development offshoring: A multi-level theoretical framework and research agenda. Journal of Global Information Management (JGIM), 14(2), pp. 52–74.

Noll, J., Beecham, S. & Richardson, I. (2010). Global software development and collaboration: barriers and solutions. ACM Inroads, 1(3), pp. 66–78.

O'Conchuir, E., Holmström, H., Agerfalk, P.J. & Fitzgerald, B. (2006). Exploring the Assumed Benefits of Global Software Development. In: the proceedings of International Conference on Global Software Engineering (ICGSE '06). October 16–19, 2006. Florianopolis, Brazil. pp. 159–168.

Orlikowski, W.J. & Baroudi, J.J. (1991). Studying information technology in organizations: Research approaches and assumptions. Information systems research, 2(1), pp. 1–28.

Paasivaara, M., Durasiewicz, S. & Lassenius, C. (2008a). Distributed agile development: Using scrum in a large project. In: the proceedings of International Conference on Global Software Development (ICGSE'08) August 17–20, 2008. Bangalore, India. pp. 87–95.

Paasivaara, M., Durasiewicz, S. & Lassenius, C. (2008b). Using scrum in a globally distributed project: a case study. Software Process: Improvement and Practice, 13(6), pp. 527–544.

Paasivaara, M. & Lassenius, C. (2003). Collaboration practices in global inter-organizational software development projects. Software Process Improvement and Practice, 8(4), pp. 183–199.

Palmer, S.R. & Felsing, M. (2001). A practical guide to feature-driven development. Pearson Education.

Parnas, D. & Clements, P. (1986). A rational design process: How and why to fake it. IEEE Transactions on Software Engineering, 12(2), pp. 80–100.

Perry, D.E., Staudenmayer, N.A. & Votta, L.G. (1994). People, organizations, and process improvement. IEEE Software, 11(4), pp. 36–45.

Petersen, K. & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. Journal of Systems and Software, 82(9), pp. 1479–1490.

Petersen, K., Wohlin, C. & Baca, D. (2009). The waterfall model in large-scale development. In: the proceedings of the10th International Conference on Product-Focused Software Process Improvement (PROFES2009). June 15–17, 2009. Oulu, Finland. pp. 386–400.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P. & Still, J. (2008). The impact of agile practices on communication in software development. Empirical Software Engineering, 13(3), pp. 303–337.

Pikkarainen, M., Korkala, M., Kääriainen, J. & Välimäki, A. (2011). Practices for efficient customer collaboration in innovation – insights from the Finnish industry. International Journal of Technology Marketing, 6(1), pp. 17–35.

Poppendieck, M. & Poppendieck, T. (2007). Implementing Lean Software Development: From Concept to Cash. Addison-Wesley Professional.

Poppendieck, M. & Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit. Addison-Wesley, Upper Saddle River, NJ, USA.

Prikladnicki, R., Audy, J. & Evaristo, R. (2003). Distributed Software Development: Toward an understanding of the relationship between project team, users and customers. In: the proceedings of 5th International Conference on Enterprise Information Systems (ICEIS2003). April 23–26, 2003. Angers, France. pp. 417–423.

Prowell, S.J., Trammell, C.J., Linger, R.C. & Poore, J.H. (1999). Cleanroom software engineering: technology and process. Pearson Education.

Qumer, A. & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Information and Software Technology, 50(4), pp. 280–295.

Qumer, A. & Henderson-Sellers, B. (2006). Measuring agility and adaptability of agile methods: a 4-dimensional analytical tool. In: the proceedings of IADIS International Conference on Applied Computing (AC'2006) February 25–28, 2006. San Sebastian, Spain. pp. 503–507.

Ramasubbu, N. & Balan, R.K. (2009). The impact of process choice in high maturity environments: An empirical analysis. In: the proceedings of the 31st IEEE International Conferences on Software Engineering (ICSE 2009). May 16–24, 2009. Vancouver, British Columbia, Canada. pp. 529–539.

Ramesh, B., Cao, L., Mohan, K. & Xu, P. (2006). Can distributed software development be agile? Communications of the ACM, 49(10), pp. 41–46.

Rizwan Jameel Qureshi, M. (2012). Agile software development methodology for medium and large projects. IET Software, 6(4), pp. 358–363.

Robert, L.P. & Dennis, A.R. (2005). Paradox of richness: A cognitive model of media choice. IEEE Transactions on Professional Communication, 48(1), pp. 10–21.

Robson, C. (2002). Real World Research, 2nd edn. Blackwell Oxford, UK.

Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012). Survey on agile and lean usage in Finnish software industry. In: the Proceedings of the ACM-

IEEE international symposium on Empirical software engineering and measurement (ESEM'12). September 19–20, 2012. Lund, Sweden. pp. 139–148.

Rogers, E.M. (1986). Communication Technology: The New Media in Society. The Free Press, New York.

Royce, W. (1990). Pragmatic quality metrics for evolutionary software development models. In: the proceedings of the Conference on TRI-ADA'90. December 3–6, 1990. Baltimore, MD, USA. pp. 551–565

Royce, W. W. (1970). Managing the Development of Large Software Systems. In: The proceedings of the IEEE WESCON. August 1970. San Francisco, CA, USA. pp. 328–339.

Runeson, P. & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), pp. 131–164.

Saeki, M. (1995). Communication, Collaboration and Cooperation in Software Development – How Should We Support Group Work In Software Development? In: the proceedings of the 2nd Asia-Pacific Software Engineering Conference (APSEC'95). December 6–9, 1995. Brisbane, Queensland, Australia pp. 12–20.

Sahay, S. (2003). Global software alliances: the challenge of 'standardization'. Scandinavian Journal of Information Systems, 15(1), pp. 3–21.

Salas, E., Sims, D.E. & Burke, C.S. (2005). Is there a "big five" in teamwork? Small Group Research, 35(5), pp. 555–599.

Salo, O. & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. IET Software, 2(1), pp. 58–64.

Sarker, S. & Sahay, S. (2004). Implications of space and time for distributed work: an interpretive study of US–Norwegian systems development teams. European Journal of Information Systems, 13(1), pp. 3–20.

Sarker, S. & Sarker, S. (2009). Exploring Agility in Distributed Information Systems Development Teams: an Interpretive Study in an Offshoring Context. Information Systems Research, 20(3), pp. 440–461.

Schwaber, K. (2004). Agile Project Management with Scrum. Microsoft Press, USA.

Schwaber, K. & Beedle, M. (2002). Agile Software Development with Scrum. Prentice-Hall, Upper Saddle River, New Jersey, USA.

Seaman, C.B. (2002). Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering, 25(4), pp. 557–572.

Shannon, C.E. (1948). A mathematical theory of communication. The Bell System Technical Journal, 27(3), pp. 379–423.

Shrivastava, S.V. & Date, H. (2010). Distributed Agile Software Development: A Review. Journal of Computer Science and Engineering, 1(1), pp. 10–17.

Sillitti, A., Ceschi, M., Russo, B. & Succi, G. (2005). Managing uncertainty in requirements: a survey in documentation-driven and agile companies. In: the proceedings of the 11th IEEE International Symposium on Software Metrics, September 19–22, 2005. Como, Italy. pp. 10–17.

Siniaalto, M. & Abrahamsson, P. (2007). A Comparative Case Study on the Impacts of Test-Driven Development on Program Design and Test Coverage. In: the proceedings of the First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). September 20–21, 2007. Madrid, Spain. pp. 275–284.

Siniaalto, M. & Abrahamsson, P. (2008). Does Test-Driven Development Improve the Program Code? Alarming Results from a Comparative Case Study. In *Balancing Agility and Formalism in Software Engineering*, eds. B. Meyer, J.R. Nawrocki & B. Walter, Springer. pp. 143–156.

Šmite, D. (2006). Global Software Development Projects in One of the Biggest Companies in Latvia: Is Geographical Distribution a Problem? Software Process Improvement and Practice, 11, pp. 61–76.

Šmite, D., Moe, N.B. & Ågerfalk, P.J. (2010a). Agility Across Time and Space: Summing up and Planning for the Future. In *Agility Across Time and Space – Implementing Agile Methods in Global Software Projects*, eds. D. Šmite, N.B. Moe & P. J. Åkerfalk, Springer. pp. 333–337.

Šmite, D., Moe, N.B. & Ågerfalk, P.J. (2010b). Fundamentals of Agile Distributed Software Development. In *Agility Across Time and Space – Implementing Agile Methods in Global Software Projects*, eds. D. Šmite, N.B. Moe & P. J. Åkerfalk, Springer. pp. 3–7.

Šmite, D., Wohlin, C., Gorschek, T. & Feldt, R. (2010c). Empirical evidence in global software engineering: a systematic review. Empirical Software Engineering, 15(1), pp. 91–118.

Solinski, A. & Petersen, K. (2014). Prioritizing agile benefits and limitations in relation to practice usage. Software Quality Journal, September 2014, pp. 1–36.

Sommerville, I. (1996). Software process models. ACM Computing Surveys (CSUR), 28(1), pp. 269–271.

Stake, R. (1995). The Art of Case Research. Sage Publications, Thousand Oaks, CA, USA.

Stankovic, D., Nikolic, V., Djordjevic, M. & Cao, D. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. Journal of Systems and Software, 86(6), pp. 1663–1678.

Stapleton, J. (1997). DSDM, dynamic systems development method: the method in practice. Addison-Wesley Professional.

Stavru, S. (2014). A critical examination of recent industrial surveys on agile method usage. Journal of Systems and Software, vol 94, August 2014, pp. 87–97.

Stol, K., Avgeriou, P., Babar, M.A., Lucas, Y. & Fitzgerald, B. (2014). Key factors for adopting inner source. ACM Transactions on Software Engineering and Methodology (TOSEM), 23(2), pp. 18.

Stol, K. & Fitzgerald, B. (2013). Uncovering theories in software engineering. In: the proceedings of 2013 2nd SEMAT Workshop on a General Theory of Software Engineering (GTSE 2013). May 26, 2013. San Francisco, CA, USA. pp. 5–14.

Summers, M. (2008). Insights into an Agile adventure with offshore partners. In: the proceedings of AGILE 2008 Conference. August 4–8, 2008, Toronto, Ontario, Canada. pp. 333–338.

Sureshchandra, K. & Shrinivasavadhani, J. (2008). Adopting Agile in Distributed Development. In: the proceedings of International Conference on Global Software Development (ICGSE'08) August 17–20, 2008. Bangalore, India. pp. 217–221.

Susman, G. I. & Evered, R. D. (1978). An Assessment of the Scientific Merits of Action Research. Administrative Science Quarterly, 23, pp. 582–603.

Sutherland, J., Viktorov, A., Blount, J. & Puntikov, N. (2007). Distributed scrum: Agile project management with outsourced development teams. In: the Proceedings of 40th Annual Hawaii International Conference on System Sciences (HICSS 2007), January 3–6, 2007. Waikoloa, Big Island, HI, USA. pp. 274–284.

Sutherland, J., Schoonheim, G. & Rijk, M. (2009). Fully Distributed Scrum: Replicating Local Productivity and Quality with Offshore Teams. In: the pro-

ceedings of 42nd Hawaii International Conference on System Sciences (HICSS '09). January 5–8, 2009. Hilton Waikoloa Village Resort, Big Island, HI, USA. pp. 1–8.

Teasley, S.D., Covi, L.A., Krishnan, M.S. & Olson, J.S. (2002). Rapid Software Development Through Team Collocation. IEEE Transactions on Software Engineering, 28(7), pp. 671–683.

Therrien, E. (2008). Overcoming the challenges of building a distributed agile organization. In: the proceedings of AGILE 2008 Conference. August 4–8, 2008, Toronto, Ontario, Canada. pp. 358–372.

Turk, D., France, R. & Rumpe, B. (2005). Assumptions Underlying Agile Software-Development Processes. Journal of Database Management, 16(4), pp. 62–87.

Uy, E. & Ioannou, N. (2008). Growing and sustaining an offshore Scrum engagement. In: the proceedings of AGILE 2008 Conference. August 4–8, 2008, Toronto, Ontario, Canada. pp. 345–350.

van Deursen, A. (2001). Customer Involvement in Extreme Programming: XP2001 Workshop Report. ACM Software Engineering Notes. Nov. 2001, pp. 70–73.

van Oosterhout, M., Waarts, E. & Van Hillegersberg, J. (2006). Change factors requiring agility and implications for IT. European Journal of Information Systems, 15(2), pp. 132–145.

Vax, M. & Michaud, S. (2008). Distributed agile: Growing a practice together. In: the proceedings of AGILE 2008 Conference. August 4–8, 2008, Toronto, Ontario, Canada. pp. 310–314.

Wallach, E.J. (1983). Individuals and Organizations: The Cultural Match. Training and development journal, 37(2), pp. 29–36.

Walsham, G. (1995). Interpretive Case Studies in IS Research: Nature and Method. European Journal of Information Systems, 4(2), pp. 74–81.

Wang, X., Conboy, K. & Cawley, O. (2012). “Leagile” software development: An experience report analysis of the application of lean approaches in agile software development. Journal of Systems and Software, 85(6), pp. 1287–1299.

West, D., Grant, T., Gerush, M. & D'Silva, D. (2010). Agile Development: Mainstream Adoption Has Changed Agility. Forrester Research.

Williams, L. & Cockburn, A. (2003). Guest Editors' Introduction: Agile Software Development: It's about Feedback and Change. Computer, 36(6), pp. 43.

Williams, W. & Stout, M. (2008). Colossal, scattered, and chaotic (planning with a large distributed team). In: the proceedings of AGILE 2008 Conference. August 4–8, 2008, Toronto, Ontario, Canada. pp. 356–361.

Wolff, J.G. (1989). The management of risk in system development: Project SP and the New Spiral Model. Software Engineering Journal, 4(3), pp. 134–142.

Wolff, S. (2012). Scrum goes formal: Agile methods for safety-critical systems. In: the proceedings of 2012 Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA). June 2, 2012. Zürich, Switzerland. pp. 23–29.

Womack, J.P. & Jones, D.T. (1996). Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Simon & Schuster, New York, NY, USA.

Yin, R.K. (2003). Case Study Research, Design and Methods. 3rd edn. Sage Publications, Beverly Hills, CA, USA.

Yin, R.K. (1994). Case Study Research Design and Methods. Sage Publications, Thousand Oaks, CA.

PAPER I

Extreme Programming: Reassessing the Requirements Management Process for an Offsite Customer

Proceedings of the 11th European Conference on
Software Process Improvement (EuroSPI 2004).
November 10–12, 2004. Trondheim, Norway. Pp. 12–22.

PAPER II

A Case Study on the Impact of Customer Communication on Defects in Agile Software Development

Proceedings of AGILE 2006 Conference. July 23–28, 2006.
Minneapolis, MN, USA. Pp. 76–88.

A Case Study on the Impact of Customer Communication on Defects in Agile Software Development.

Mikko Korkala[1], Pekka Abrahamsson[2] and Pekka Kyllönen[2]

[1]*Department of Information Processing Science*
P.O.Box 3000, FIN-90014 University of Oulu, Finland
[2]*VTT Technical Research Centre of Finland*
P.O.Box 1100, FIN-90571, Oulu, Finland
Mikko.Korkala@oulu.fi; *Pekka.Abrahamsson@vtt.fi;pekka.kyllonen@vtt.fi*

Abstract

Effective communication and feedback are crucial in agile development. Extreme Programming (XP) embraces both communication and feedback as interdependent process values which are essential for projects to achieve successful results. Our research presents the empirical results from four different case studies. Three case studies had partially onsite customers and one had an onsite customer. The case studies used face-to-face communication to different extents along with email and telephone to manage customer-developer communication inside the development iterations. Our results indicate that an increased reliance on less informative communication channels results in higher defect rates. These results suggest that the selection of communication methods, to be used inside development iterations, should be a factor of considerable importance to agile organizations working with partially available customers. This paper also proposes some guidelines for selecting proper communication methods.

1. Introduction

Misunderstood or erroneous requirements are often the reason why software systems are late and unreliable [1]. Literature and practice show that agile methods work with unstable requirements which are managed through short development cycles and efficient communication (e.g. [2, 3]). Agile development approaches tend to rely on effective personal communication and collaboration between the project participants [2, 4].

In fact, two agile principles (http://agilemanifesto.org/principles.html) directly emphasize the importance of customer-developer communication. Face-to-face communication is identified as the most efficient means of communication between the participants. Moreover, the daily collaborative work of business people and developers demands efficient verbal communication between the customer and the developers. In addition to communication, feedback is also crucial in agile development. The most popular agile method Extreme Programming (XP) embraces both communication and feedback as interdependent process values. Previously, XP went to extremes with customer communication and feedback by proposing an *Onsite Customer*, a real person constantly available for information [3]. Although some authors (e.g. [5-8]), have found weaknesses in this practice, on the positive side, having an onsite customer when possible enables efficient communication and feedback since the customer is always and easily available for information. Even though the practice has its advantages, it is not always possible to have the customer constantly onsite due to various reasons (e.g. the customers are too valuable or too remote to be onsite) [9]. The *Onsite Customer* has now been replaced with the *Real Customer Involvement* practice which does not assume fulltime customer presence, but suggests that the customer can participate in weekly and quarterly meetings [10].

As demonstrated above, the importance of communication has been shown to be paramount in agile development. Less however is known about the implications of communication and feedback in this context. The purpose of this paper is therefore twofold. First, this paper explores the impact of customer communication and feedback on software defect rates. Second, it briefly discusses the selection

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

of customer communication channels based on the existing literature.

The empirical material is based on four case studies. From one of the case studies, the researchers were able to capture reliable metrics about the defect division and the times required to fix those defects. Defect follow-up provides a means for increasing our understanding of how well a team has understood the customer's intentions. The other three case studies, including one onsite customer project, are used to show trends in the development of defect densities. These cases applied different communication channels, varying in their richness. The richness of the communication channels is observed from the viewpoint of the Media Richness Theory (MRT) [11, 12], and personal suggestions on the information richness of different communication mediums by Ambler [13] and Cockburn [4].

We claim that effective and rich mid-iteration communication[1] is a highly important factor in cases where rich customer communication is not always available, and the lack of high-bandwidth (e.g. face-to-face) communication and an increasing reliance on leaner communication channels results in higher defect rates, despite careful face-to-face requirements analysis.

The principal findings of the study are as follows: It was found that an average of 25.3% of all tasks involved fixing defects in the three non-onsite customer projects. The case proving accurate metrics about the defects indicated that an average of 62.6% of all defects could have been avoided by either having an onsite customer or a more effective mid-iteration communication mechanism. These results indicate that the selection of a mid-iteration communication medium, especially, must be given significant attention in agile development efforts with partially available customers. Some suggestions to overcome these challenges will be presented in this paper.

This paper is organized as follows: section 2 discusses the relevant communication literature, focusing on the MRT and communication in agile context; Section 3 explains the research settings; the empirical research results and our discussion on agile communication are in section 4; and the conclusions are drawn in section 5.

[1] Mid-iteration communication is used for the customer-developer communication taking place inside any particular iteration.

2. Related literature

Effective communication is an essential success factor for any software development effort (e.g. [3, 14, 15], but according to Cockburn [4] it is far from being an easy task. Sometimes the idea to be expressed is not even fully conscious, and for the communication to be effective the participants must have a shared experience of the subject that is being discussed. This shared experience is needed for the participants to compensate for the gaps in communications. [4]

Section 2.1 discusses the Media Richness Theory which, despite criticism, is possibly the most well-known theory of media selection. Section 2.2 focuses on communication in the context of agile software development.

2.1. Media richness theory (MRT)

Media Richness Theory (MRT) [11, 12] is perhaps the most prominent theory of media selection, even though it has been widely criticized (e.g. [16-18]). The theory suggests that matching the task needs to mediums ability to convey information will improve task performance and that certain media are better capable of transmitting uncertain or equivocal information [11, 12].

Uncertainty is interpreted as the lack of information [19], while equivocality means that there are multiple, possibly conflicting interpretations for the information [16]. The values of these variables can be classified as high or low [11]. According to Daft, Lengel and Trevino, uncertainty leads to data acquisition whereas equivocality results in the exchange of subjective views, definition of the problem and resolution of disagreements [12]. Daft and Lengel [11] suggest that equivocal tasks should be managed through rich communication channels, while leaner channels are effective for processing standard data and well understood messages.

The richness of different communication mediums is based on a blend of four criteria [12]. The characterization of these variables is complemented by the perceptions of online media by Graveline, Geisler and Danchak [20], as follows:

1) Feedback, instant feedback enables asking questions and making corrections. The capability of feedback relates to synchronicity of feedback; online media are either synchronous or asynchronous [20];

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

2) Multiple cues, different cues (e.g. body language, voice inflection) can be a part of the message. Some online media are capable of transmitting multiple cues (e.g. videoconferencing), some are primarily single channelled (e.g. email) [20];

3) Language variety, refers to the use of different languages, such as numbers which convey precision and natural language conveying the understanding of a broader set of ideas. Written or typed language provides more precise language [20];

4) Personal focus, A message will be conveyed more fully when personal feelings are infusing the communication. However, sometimes lesser emotional content can be better. Cockburn [4] provides an example of a project leader and her team who used telephones because she was too aggressive with her emotions in person.

MRT suggest that the richest communication channel enables instant feedback, is capable of transmitting multiple cues, has a high language variety and it can transmit emotional contents.

Based on [12], the following summary of the richness of different communication channels can be made. Face-to-face is the richest form of communication since it enables instant feedback and it is able to transmit multiple cues [12]. Videoconferencing is capable of sending both video and audio, and it enables fast feedback [12]. Videoconferencing is however a somewhat leaner medium than face-to-face and it restricts some visual cues, but it has more information capacity than telephone [12]. Email has capabilities similar to telephone, but the cues are filtered out [12].

As mentioned above, MRT has been criticized by several authors. For example, Dennis and Valacich [16] conclude that the richest medium is defined by the situation in which it is used. The richness of medium depends on its ability to provide the capabilities needed by the situation. These capabilities include the individuals, tasks and the social context in which they interact [16]. They also state that media is not monolithic, since media are capable of possessing different levels of communication depending on their usage (e.g. video and audio files can be attached to email messages) [16].

2.2. Communication in agile development

Agile methods work with volatile requirements and embrace personal communication between the participants [2]. Since the level of ambiguity can be considered high in the form of unstable requirements, it seems that personal communication is a natural choice (and in line with MRT) for agile development. Based on the existing literature, Kraut & Streeter [21] summarized the characteristics of informal communication. They state that informal communication is the primary method of exchanging information within research and development organizations [21]. Also in many domains, the physical proximity of the information source is also important, and transmitting the information through informal, interpersonal communication is valuable especially as development and research tasks become more uncertain [21]. Also Cockburn [4] agrees that informal communication between the stakeholders and their physical proximity are important factors. Personal, verbal communication has also been supported e.g. by the following arguments: according to Juric [22] verbal communication enables defect detection early in the development process, and personal contact with the customer causes less distortion to the information [23]. Melnik and Maurer [23] compared the differences between traditional and agile development. Since in traditional development the information travels through a long chain of people, it becomes distorted and some of it is lost. Agile development embraces personal communication with the customer, which reduces both information loss and mutation. [23] Informal and personal communication is thus highly efficient, but it has to be maintained since according to Wake [24], the lack of high-bandwidth communication leads to a loose feedback mechanism. This makes the steering of a project become less responsive, which in turn demands more effort to get the project back on course [24].

Despite its usefulness, verbal communication has also been criticized by e.g. Finsterwalder [5] who has pointed out the pitfalls in verbal communication when discussing Extreme Programming, according to Finsterwalder verbal communication is not always well thought-out and the contents of the conversations can be forgotten during time.

In addition to communication, feedback is another essential success factor (e.g. Wake [24]). The most popular agile method Extreme Programming (XP) embraces communication and feedback as interdependent process values. The *Communication* value of XP requires that the communication must be maximized between the stakeholders through direct, interpersonal communication. *Feedback* value, for its part, requires that the contact with the customer is

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

close. These values do not exist in isolation but are strongly related; without communication the feedback cannot exist and vice versa [10]. Previously XP defined *On-site customer* practice as requiring the customer's fulltime presence with the team in a same shared workspace [3]. Even though this practice has been found problematic (e.g. [5-8]), it enabled effective communication and instant feedback.

The *On-site customer* has now been replaced by less demanding *Real Customer Involvement* practice which does not explicitly require customer's constant presence. The customer who can contribute to weekly and quarterly planning is still an essential part of the team, and team should have close contact with him [10] . Since the customer is not any longer required for fulltime presence, the already important roles of communication and feedback become even more essential. Thus it is essential that the communication and feedback mechanisms should receive special attention in agile development.

Gottesdiener [25] has defined a framework for holding requirements workshops with various time and place combinations. This framework can be applied to agile development, thus indicating that the teams can select a proper communication channel from considerably wide collection of different media. In this work, time can be either synchronous with every participant present at the same time or asynchronous. Place can be either co-located or distributed when participants are at different locations.

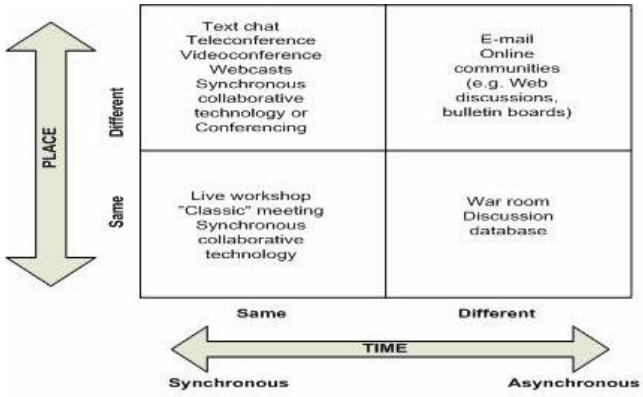

Figure 1. Requirements workshop techniques for different time-place configurations, reproduced from [25].

McLuhan [26] has used the terms "hot" and "cool" when discussing media. McLuhan argues that "cool" media require a higher level of participation from users than "hot" media, and because of the lack of information, much has to be filled in by the users. McLuhan also adds that the use of "hot" and "cold" media has to be adapted to the underlying situation [26]. Ambler and Cockburn [4] have also adapted the use of "hot" and "cold" communication media. They argue that "hot" communication channels provide more information than "cold" media. The following figure is Ambler's [13] modification of Cockburn's comparison of communication effectiveness in different communication media. Different communication channels have been divided into "cold" and "hot" channels based on their communication effectiveness. "Hot" communication channels are thus more efficient than "cold" media. Both Cockburn's and Ambler's work is based on personal experience and has not been scientifically evaluated.

As can be seen from the graph in Fig. 2, face-to-face communication is the "hottest" communication channel, while email is positioned in the "colder" end of the graph. This division is similar to the classification made by Daft et al in [12]. Thus, both theory and practice indicate that personal communication is the richest form of communication, while less interactive methods are less effective forms of communication.

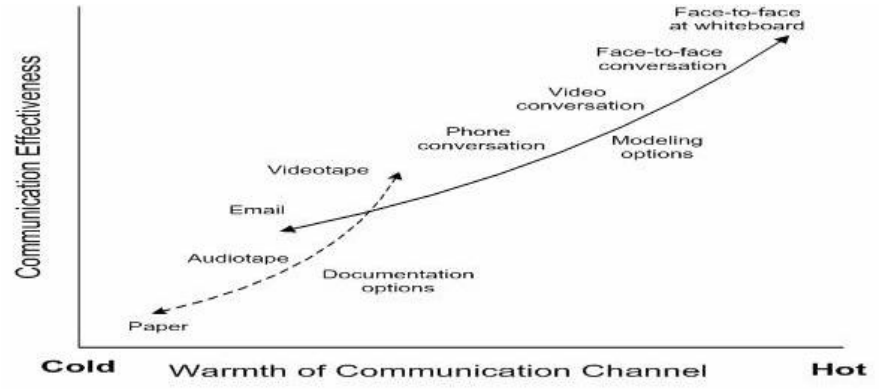

Figure 2. The communication value of different techniques, reproduced from [13].

3. Research design

This section describes the research method, data collection and analysis techniques, and research settings on which the results of this study are based. Our research covered four different case [27] studies, referred to as *Case 1, Case 2, Case 3* and *Case 4*. *Case 1* has been previously described in [6] and *Case 2* is discussed in [28]. The teams consisted of 5 to 6 developers, and in the case of *Case 3* one of the developers was a member of the customer organization. The aim of projects *Case 2, Case 3* and *Case 4* was to implement a mobile software product to the customers, while *Case 1* delivered an application for internal use. In the case of *Case 3*, the assignment was to implement a mobile version of an existing desktop application, implemented by the customer organization.

All the cases were implemented following Mobile-D™ (hereafter Mobile-D). Mobile-D is inspired by several agile methods and it is specifically designed for mobile application development. Mobile-D does not require a fulltime customer presence. Mobile-D proposes that the customer should participate personally in the development when the contents of the iterations are discussed and analysed, and during the release when the customer can provide feedback about the results of the iterations. Otherwise the customer is not onsite.

Mobile-D consists of 5 iterations. The first and the last iterations last for a week, while the duration of the remaining three is 2 weeks [29]. For the purposes of the study, the development method is only briefly described to provide background information about the research. Mobile-D is explained in greater detail in [29, 30].

Before the projects were launched, the teams (including the customers) were introduced to the philosophy and principles of agile development (Agile Manifesto, available at http://agilemanifesto.org/) and trained to use the Mobile-D. The following table summarizes the characteristics of the case projects.

Table 1. Summary of the case projects.

	Case 1	Case 2	Case 3	Case 4
Iterations	5	5	5	5
Iteration span (weeks)	1+2+2+2+1	1+2+2+2+1	1+2+2+2+1	1+2+2+2+1
Number of product releases	4	4	4	4
Product type	Intranet application	Mobile application	Mobile application	Mobile application
Programming	Java	Mobile Java	Mobile Java	Mobile Java

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

	Case 1	Case 2	Case 3	Case 4
language				
Customer presence	Onsite 80%	Onsite during the beginning and the end of each iteration. Available for face-to-face communication.	Onsite during the beginning and the end of each iteration.	Onsite for the first 2 weeks. Afterwards not onsite.
Customer location	Same facility	Same facility	Same city	Different city
Number of customers	1	3	1	1
Face-to-face communication level	High	On demand	Periodical	Low
Mid-iteration communication media used	Face-to-face	Face-to-face, email	Email	Email and telephone

During our research the utilization of face-to-face communication varied as follows: *High* (onsite customer), *On demand* (face-to-face communication available when needed), *Periodical* (face-to-face communication applied at the beginning and the end of an development iteration) and *Low* (customer onsite for the first two weeks, afterwards face-to-face communication was not applied or available). *Case 1* had an onsite customer, and during *Case 2* the customer was required to be onsite during the Planning Day and Release Day activities of the Mobile-D [29]. These phases concentrate on deciding and analyzing the contents of each development iteration and release, when the results of the iteration are presented to the customer. During these sessions, the customer was available for face-to-face conversation. The mid-iteration communication was managed via email. The customer was, however, available for face-to-face communication on demand, so personal communication and feedback were available also during the iterations. *Case 3* was similar to *Case 2,* excluding the mid-iteration face-to-face communication.

During *Case 4* the customer was constantly available for the first two weeks for guidance and feedback. Thereafter, the communication was managed entirely by email and telephone.

The task data was collected from each case and the defect fixing tasks isolated. The amount of defect-fixing tasks, from all the tasks, were calculated. The teams were required to record the time needed to fix the defects, but this was not properly executed except in the case of *Case 4*. Table 2 summarizes the defect data available from the case projects.

Thus the results from *Case 1, Case 2* and *Case 3* can be used only to show trends in the amount of defect densities. It must be noticed that the defect data from the first release of *Case 2* is missing. In addition to the quantitative research data, qualitative data was also used. This data is based on onsite observations, a research diary and interviews.

Table 2. Available defect data from the case projects.

	Case 1	Case 2	Case 3	Case 4
Defect data	X	X	X	X
Defect fixing times	-	-	-	X

4. Empirical research results

In 4.1 the findings of the study are presented. In 4.2 some techniques for customer communication are discussed based on the existing literature.

4.1. Impact of customer communication on software defects

Both theory and practice have shown that rich face-to-face communication is effective, and its application proved itself very useful in our research. The requirements analysis was conducted face-to-face with the customers and the conversations were further clarified by lightweight paper-prototyping. This approach initiated very lively discussions, thus enabling rich communication between the team and the customers. When compared to the suggestions of Media Richness Theory [11, 12] and personal experiences of both Cockburn [4] and Ambler [13], this finding is not surprising.

In spite of the wide array of different communication methods available for managing mid-iteration communication, only two of the leanest channels described in chapter 2 were used in non-onsite customer cases. *Case 2* is an exception, since

the team was able to discuss with the customer in person even during the development period. Only email (and in *Case 4* also telephone) was used to exchange information between the customers and the rest of the development team during the development. These media were selected, even though the customers were encouraged to choose richer channels. During *Case 2,* the customers specifically asked for simple reports since they were missing the "big picture" from technically oriented reports [28]. The reasons why the customers wanted to use simple reports in the mid-iteration communication during other cases remains unclear. The customers' inexperience in agile development could be one possible explanation. If the customers are not fully aware of the nature of agile development, the importance of communication and feedback can go unnoticed. During *Case 2,* the customer stated the initial requirements and left the team alone to complete the product without any support or feedback. This situation is described in more detail in [28]. In addition, the findings of Kinney and Watson [18] are interesting. They report that email was found to be an appropriate medium for most communication tasks, including both equivocal tasks, which according to Media Richness Theory should be handled through a richer medium, and non-equivocal tasks. We have some evidence that supports this. When asked about media selection, the customer in *Case 3* responded "*Emails will do just fine, they are enough"*. Thus it seems that the customers were comfortable with simple email reports, and did not want to change the mid-iteration communication medium in spite of the increasing amount of defects.

Even though the customers were pleased with simple reports, which have been found to provide an excellent view of a project's progress at very low cost [28, 31], it is evident that they do not provide enough information for crucial customer feedback. The following excerpt is from an actual status report sent to the customer in *Case 3*[2]. The report describes the number and name of the stories and their status (in this case either STARTED or DONE) and some brief explanation of their functionalities[3].

"*Story 9: Bug Fixes of R2 - DONE*
- Lots of work with scrolling
Story 10: Redesign Architecture – STARTED
- Basically finished, move 1 task to the next release
Story 11: Orders Are Shown in OrderListScreen - DONE
Story 12: Enhancements to SalesItemScreen - STARTED
- Week Level still in progress
Story 13: OrderInfoScreen (Info of One Order) - DONE
Story 14: Search Feature – DONE"

This report indicates that several stories were completed, but it is obvious that the simplicity of the report could not enable the customer to give feedback that could have been used to steer the development. The analysis of the email correspondence between the teams and the customers supports this claim. The customers did not give any feedback based on the reports during *Case 2* and *Case 3*. The customer responded once to an email report explaining implemented features during *Case 4*.

Based on the report above, the customer was not able to verify that the features were in fact implemented correctly, thus making the contents of the report partly ambiguous. On the other hand, ambiguity does not exist when observing stories with the status of STARTED, since these parts are well-defined. Thus this supports the claims of MRT that ambiguous issues should be managed through rich channels, while leaner channels are adequate for well-understood subjects. Similarly, this also supports the works of Cockburn [4] and Ambler [13].

The customers' inability to give feedback based on the reports further supports the claim by Wake [24], who states that the lack of high bandwidth communication leads to a loose feedback mechanism, which in turn requires more effort to get the project back on course.

The following table summarizes the times in minutes required for defect corrections during *Case 4.* As mentioned in section 3, there is not sufficient data available from other cases. The teams, excluding *Case 4* did not record a defect log in sufficient detail. The defects were at first categorized into customer independent and customer dependent defects based on their descriptions. Customer independent defects are such that can be traced back to developers or to the development platform. Customer dependent defects are defects which could have been avoided by either having constant customer communication and feedback (onsite customer) or utilizing a rich mid-iteration communication method. Based on the defect data analysis, customer independent errors were further divided into two categories. They were either caused by developers or, as in one case, the defect was caused by the underlying Java virtual machine. Similarly the customer dependent defects were divided into cosmetic (e.g. the user interface was not what the customer expected, its elements were not as required,

[2] Even though the reports did not follow a similar format, the information contents were similar. The reports in all the cases were very simple and described what was done only on a general level.
[3] The comments are translated from Finnish.

the customer was not pleased with the colors or fonts) or they had requirement related defects (i.e. the features did not work as expected).

Table 3. Defect fix times in minutes for customer independent and customer dependent defects in *Case 4*.

Release	Customer independent		Customer dependent	
#	Developer	Platform	Cosmetic	Requirement
1	310	-	-	-
2	150	-	70	355
3	230	-	160	595
4	30	20	-	30
Subtotal	**720**	**20**	**230**	**980**
Total	**740**		**1210**	

It can be interpreted from the above data that the cost of fixing customer dependent defects was ~1.6 times larger in terms of time when compared to customer independent defects. Further analysis indicated that the percentage of defects in the above table was distributed as follows: *developer*: 37%, *platform*: 1%, *cosmetic*: 12% and *requirements*: 50%. When releases were reviewed separately the amount of customer dependent defects was distributed as set out in Table 4.

Table 4. The proportion of customer dependent defects/release in *Case 4*.

Release #	% of all defects
1	0
2	73,9
3	76,6
4	37,5

The contents of the first release during all the cases concentrated on the technical infrastructure. In *Case 4* the only implemented feature on which the customer was able to give feedback to the team was login functionality. During this particular iteration, all the defects were caused by the carelessness of the developers.

Clearly, the rate of customer dependent defects was very high during the second and third release and remained considerably high also in the final release. Since face-to-face communication was applied only during the first two weeks, the communication was managed entirely through lean communication channels thereafter. This further supports the MRT's claim that lean channels are not suitable for managing ambiguous issues. In summary, during *Case 4,* 62.6% of software defects were caused by the lack of efficient customer communication and feedback.

Our other findings, even though not as precise, also support this viewpoint. The following figure shows the percentage of all defect-fixing tasks in each project. The average percentage of defect-fixing tasks was 6.3% in *Case 1* and we can assume that all the defects were in this case developer-oriented. In *Case 2* the percentage of the defect-fixing tasks was reasonably low on average, whereas it increased continuously in *Case 3* and *Case 4*. The last release of *Case 4* concentrated solely on defect-fixing, because all the other scheduled functionalities were cancelled.

For more precise information about the defect task percentages, the following table summarizes the averages for defect-fixing tasks in each case project.

Table 5. The averages for defect-fixing tasks in the case projects.

Project	Utilization of face-to-face communication	Average of defect fixing tasks (%)
Case 1	High	6,3
Case 2	On demand	9,4
Case 3	Periodical	24,7
Case 4	Low	41,8

When combining the figures from *Case 2, Case 3* and *Case 4,* 25.3% share of all the tasks were defect fixing tasks in non-onsite customer projects. On the bases of figure 3 and table 5, it can be seen that the averages for defect-fixing tasks in *Case 1* and *Case 2,* and their variation in different releases, are very similar to each other. Even though the customers were not constantly onsite during *Case 2*, and the

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

mid-iteration communication was managed mainly by email, they were located in the same facility (*"He [one of the customers] was almost always there in the room next to us" [Case 2]*) and thus easily available for communication and feedback. Thus the amount of face-to-face communication and feedback was higher than in the other non-onsite customer cases. *Case 2* had a beneficial situation since the number of customers was higher in *Case 2* than in the other cases, and at least one the customers was always available at the beginning and the end of the iterations for face-to-face communication and feedback. Thus the customer communication was regular, while in *Case 3* the customer was not always able to join the scheduled meetings which then had to be postponed.

A more alarming trend can be observed when comparing *Case 3* and *Case 4*. These cases applied lower levels of face-to-face communication and relied more on leaner communication channels, not suitable for ambiguous issues. This backfired in higher defect rates. *Case 3* used *Periodical* face-to-face communication and suffered from a 24.7% defect ratio, while *Case 4* using a *Low* level of face-to-face communication had a defect rate of 41.8%, with 62.2% of all defects being customer-related. The development of the defect ratios during these cases also follows a similar pattern. The defect rates were constantly increasing as the projects progressed, leaving less time to implement new features or enhance the existing features. Thus our research indicates that the lower adoption of high-bandwidth communication and increasing reliance on lean communication techniques result in higher defect rates.

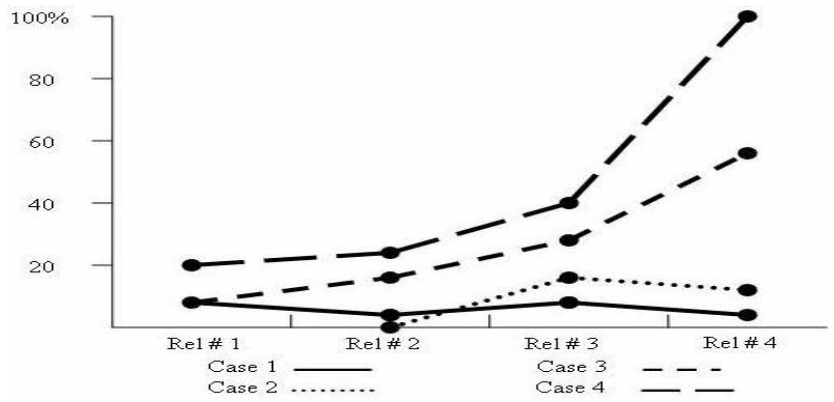

Figure 3. The percentage of defect-fixing tasks in the case projects.

These findings suggest that effective customer communication and selection of efficient communication media, especially during mid-iteration, should receive proper attention in agile development.

4.2. Suggestions for customer communication

As discussed in chapter 2, there is a large amount of different communication mechanisms that can be integrated into agile development as well. The focus of this paper is however out of providing a systematic solution to customer communication in agile development, but based on existing literature described here and in section 2, some suggestions can be made.

Face-to-face communication is the default communication method in agile development. It has proven itself effective, so it should be applied whenever possible. **Videoconferencing** comes second in richness to face-to-face and e.g. Wallace et al. [32] have suggested its utilization in user story explanation when the customers are remote. We suggest using videoconferencing in situations that need effective customer communication and feedback, where the customer is not available. The usage of videoconferencing has both positive and

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

negative effects. Sumner and Hostetler [33] have identified the following shortcomings. The difficulty of coordinating and clarifying ideas, and the time required to achieve consensus and arrive at a decision increases [33]. On the other hand, they state that computer conferencing produces better decisions, a wider range of options, greater analysis, and it creates the psychological distance needed to engage in a more open exchange of options. It is also reported in [34] that in the case of requirements negotiation, the teams communicating face-to-face did not outperform teams using computer support. **Telephone:** A telephone is not capable of transmitting visual cues, but it enables instant feedback. Telephone can be useful for example for negotiating, for example, schedules with the customer. **Email:** Emails are suitable for communicating well-understood issues.

However, it is not recommended that one blindly chooses a number of different methods for enriching the communication between the customer and the developers. Rather, the decision about communication channels should be agreed in collaboration with the different parties involved, as it also depends on the situation at hand as proposed by Dennis [16] (See also McLuhan [26]).

While the selection of the most appropriate communication method in agile development is not a focus of this paper, we believe it is not a straightforward task and it is a subject that requires further research.

5. Conclusions

Agile software development emphasizes face-to-face communication as the primary communication channel between the project participants. The existing literature has promoted the benefits of face-to-face communication, but less is known about the implications of communication and feedback in the context of agile development.

Our research in four different case studies provides empirical evidence of the effects of customer communication. Despite the considerably wide array of existing communication methods, the customers chose email, and in one case also telephone, to manage both clear and ambiguous issues inside the development iterations. Our research indicates that when the reliance on leaner communication mediums increases, the software defect rates increase accordingly. These results suggest that the selection of communication methods to be used inside development iterations must be paid significant attention by agile organizations working with partially available customers. Even though it is not the focus of this paper, some suggestions for customer communication are also discussed based on the existing literature.

We claim that a defect follow-up provides a means for increasing our understanding of how well a team has understood a customer's intentions. Yet, we do not encourage developers and customers to aim at reducing defect rates per se. We rather emphasize the importance of intense and meaningful communication. This is bound to decrease the number of misunderstandings, increase the efficiency of the work and offer improved opportunities for learning.

This paper has empirically shown that the defect rates increase due to the use of less informative communications channels in the development time. However, due to the agile principle of short iterations, this is less risky than in traditional software development where the length of an iteration may last for quite long periods. Our empirical data, however, demonstrates that there is room for improvement even in short cyclical agile development by making good use of rich communication mechanisms as we propose in this paper.

6. References

[1] L. Bernstein. (2005, Taking software requirements creation from folklore to analysis. *ACM SIGSOFT Software Engineering Notes 30(5),* pp. 3-5.

[2] P. Abrahamsson, O. Salo, J.Ronkainen and J. Warsta, *Agile Software Development Methods: Review and Analysis.* Espoo: VTT Publications, 2002, pp. 107. Available at:www.inf.vtt.fi/pdf/publications/2002/P478.pdf

[3] K. Beck, *Extreme Programming Explained: Embrace Change.* Upper Saddle River, New Jersey: Addison-Wesley, 2000.

[4] A. Cockburn, *Agile Software Development.* Indianapolis: Addison-Wesley, 2002.

[5] M. Finsterwalder, "Does XP need a professional customer", XP2001 workshop on customer involvement, in *XP2001,* Cagliari, Italy, 2001.

[6] J. Koskela and P. Abrahamsson, "On-site customer in an XP project: Empirical results from a case study," in *EuroSPI2004,* Trondheim, Norway, 2004, pp. 1-11.

[7] A. Martin, "A case study: Exploring the role of customers on extreme programming projects," Victoria University of Wellington, New Zealand, 2003.

[8] P. Abrahamsson and T. Jokela, "Usability assessment of an extreme programming project: Close co-operation with the customer does not equal to good usability," in *Profes 2004,* Kansai Science City, Japan, 2004, pp. 393-407.

[9] R. Jeffries, A. Anderson and C. Hendrickson, *Extreme Programming Installed*. Upper Saddle River, New Jersey, USA: Addison-Wesley, 2001.

[10] K. Beck and C. Andres, *Extreme Programming Explained; Embrace Change, Second Edition.* Upper Sadle River, NJ, USA: Addison-Wesley, 2005.

[11] R. L. Daft and R. J. Lengel, "Organizational Information Requirements, Media Richness and Structural Design," *Manage. Sci.,* vol. 32, pp. 554-571, 1986.

[12] R. L. Daft, R. Lengel and L. K. Trevino, "Message Equivocality, Media Selection, and Manager Performance:Implications for Information Support Systems," *MIS Quarterly,* vol. 11, pp. 355-366, 1987.

[13] S. W. Ambler, "Validating Agile Models," *Cutter IT Journal,* vol. 15, pp. 33-39, 2002.

[14] A. Edstrom, "User Influence and the Success of MIS Projects: A Contingency Approach," *Human Relations,* vol. 30, pp. 580-607, 1997.

[15] R. B. Bostrom and B. D. Thomas, "Achieving excellence in communications: A key to developing complete, accurate and shared information requirements," in Proceedings of the Twentieth Annual Computer Personnel on Research Conference, Charlottesville, Virginia, United States, 1983, pp. 1-13.

[16] A. R. Dennis and J. S. Valacich, "Rethinking media richness: Towards a theory of media synchronicity," in HICSS'99, Hawaii, United States, 1999, pp. 1017.

[17] S. Kinney and A. Dennis, "Reevaluating media richness: Cues, feedback, and task," in HICSS'94, Hawaii, United States, 1994, pp. 21-30.

[18] S. T. Kinney and R. T. Watson, "The Effect of Medium and Task on Dyadic Communication" in Proceedings of the 13th International Conference on Information Systems, Dallas, Texas, United States, 1992, pp. 107-117.

[19] R. L. Daft and K. Weick, "Toward a Model of Organizations as Interpretation Systems," *Academy of Management Review,* vol. 9, pp. 284-295, 1984.

[20] A. Graveline, C. Geisler and M. Danchak, "Teaming together apart:Emergent patterns of media use in collaboration at a distance," in Proceedings of the 2000 joint IEEE International and 18th Annual Conference on Computer Documentation, Cambridge, Massachusetts, United States, 2000, pp. 381-393.

[21] R. E. Kraut and L. A. Streeter, "Coordination in Software Development," *Communications of the ACM,* vol. 38, pp. 69-81, 1995.

[22] R. Juric. "Extreme programming and its development practices". in Proceedings of the 22nd International Conference on Information Technology Interfaces, ITI2000, Pula, Croatia, 2000, pp. 97-104.

[23] G. Melnik and F. Maurer, "Direct verbal communication as a catalyst of agile knowledge sharing," in Agile2004, Salt Lake City, Utah, United States, 2004, pp. 21-31.

[24] W. C. Wake, *Extreme Programming Explored.* Addison-Wesley, 2002.

[25] E. Gottesdiener, *Requirements by Collaboration:Workshops for defining needs. Boston:*Addison-Wesley, 2002.

[26] M. McLuhan, *Understanding Media:The Extensions of Man.* New York: McGraw Hill, 1964.

[27] R. K. Yin, *Case Study Research Design and Methods.* 2nd ed. Sage Publications, 1994

[28] M. Korkala and P. Abrahamsson, "Extreme programming: Reassessing the requirements management process for an offsite customer," in *EuroSPI 2004,* Trondheim, Norway, 2004, pp. 12-22

[29] T. Ihme and P. Abrahamsson, "Agile Architecting:The Use of Architectural Patterns in Mobile Java Applications," *IJAM,* 2005.

[30] P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen and O. Salo, "Mobile-D: An agile approach for mobile application development," in *OOPSLA 2004,*Poster session, Vancouver, Canada, 2004.

[31] M. Korkala, "Extreme programming: Introducing a requirements management process for an offsite customer," Department of Information Processing Science Research Papers Series A, University of Oulu, Finland, 2004.

[32] N. Wallace, P. Bailey and N. Ashworth, "Managing XP with multiple or remote customers," in *3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP2002),* Alghero, Italy, 2002, pp. 134-137.

[33] M. Sumner and D. Hostetler, "A comparative study of computer conferencing and face-to-face communications in systems design," in *2000 ACM SIGCPR Conference on Computer Personnel Research,* Chicago, Illinois, United States, 2000, pp. 93-99.

[34] D. E. H. Damian, A. Eberlein, M. L. G. Shaw and B. R. Gaines, "Using Different Communication Media in Requirements Negotiation," *IEEE Software,* vol. 17, pp. 28-36, 2000.

Proceedings of AGILE 2006 Conference (AGILE'06)
0-7695-2562-8/06 $20.00 © 2006 IEEE

PAPER III

Communication in Distributed Agile Development: A Case Study

Proceedings of the 33rd EUROMICRO Conference on
Software Engineering and Advanced Applications
(EUROMICRO 2007), August 28–31, 2007.
Lübeck, Germany. Pp. 203–210.

Communication in Distributed Agile Development: A Case Study

Mikko Korkala and Pekka Abrahamsson
VTT Technical Research Centre of Finland
P.O.Box 1100, FIN-90571, Oulu, Finland
Mikko.Korkala@vtt.fi; Pekka.Abrahamsson@vtt.fi

Abstract

Distributed software development is an increasingly important development approach for software companies as it brings tempting opportunities. Distributed development is already burdened with several problems and agile methods bring further challenges in the form of their reliance on verbal communication and volatile requirements. There is little empirical knowledge on distributed agile software development. We conducted two distributed agile software development case studies and compared our findings against existing recommendations about communication in distributed agile development. Our findings along with existing literature conclude that presented recommendations are worthwhile considering in distributed agile development, but with some caution. Our empirically based findings indicate that the role of a well-defined customer is the key recommendation. The lack of a well-defined customer able to meet responsibilities, as well as volatile requirements and inefficient communication, can cause severe problems even in small-scale distributed agile software development projects. Discussed recommendations are complemented with an additional recommendation.

1. Introduction

Distributed software development (DSD)[1] is becoming a common practice in modern software industry [e.g. 1], where the level of distribution can range from team members being located in the same city to those on different continents [2]. The significance of DSD has accelerated because of factors such as improving time-to-market through constant development across different time-zones, quick formation of virtual teams and the benefits of business market advantages. These needs have driven the software development efforts further towards a multi-site globally distributed environment. [1] Simultaneously, several studies have concluded that distributed enterprises are risky [e.g. 3-5]. For example, communication and coordination, software quality, schedule overruns and exceeded costs are some of the problems troubling both single-site and distributed software projects. However, the extent of the problem in the case of DSD seems to be so complex that a thorough understanding of it has not yet been defined. [3, 4] Several studies agree that communication is a particularly important issue in distributed agile development, [e.g. 5-7]. Agile methods rely on volatile requirements that are managed through efficient verbal communication [8] and thus agile software development methods pose their own challenges to the field of DSD.

In order to tackle the problems of DSD, several different techniques have been proposed. These techniques range from using different tools, such as instant messaging [9], videoconferencing [10] and whiteboard software [5] to a set of more general recommendations [5]. We conducted two different case studies with different levels of distribution ranging from the customer being in the same city, to one with a geographical distribution of 600 kilometers within the same country. Therefore, cultural differences were not an issue in these cases. We compared our findings against the recommendations of Layman et al. [5] and provide more insight on their application based on our empirical findings and the existing literature. Even though we were able to evaluate only three recommendations out of the existing four, our contribution provides valuable insight into conducting distributed agile projects. Our results further emphasize the critical role of effective communication, indicating that inefficient and irregular communication in conjunction with volatile requirements can cause severe problems even in very small-scale agile projects. However, it seems that effective communication is not the key. Our cases suggest that having a well-defined customer[2] is the key recommendation affecting to recommendations about having a Development Manager [5] and using asynchronous communication channels. As ineffective customer collaboration may render the other recommendations redundant, effective customer collaboration seems to be a key factor for successful distributed agile development. In addition, we complement the existing recommendations by introducing

[1] The literature discusses both Distributed Software Development (DSD) and Global Software Development (GSD). In this research, DSD is used to cover both terms if not explicitly stated otherwise.

[2] In this study the term means those persons responsible for product requirements and prioritization, i.e. the case projects' customers.

33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007)
0-7695-2977-1/07 $25.00 © 2007 IEEE

an additional recommendation: i.e. *enable and support direct communication between the developers*. Unexpectedly, the teams in the second case were not allowed to communicate directly with each other. To compensate, a management-led communication channel was established to balance the communication flow, which gave poor results. This factor severely blocked the progress of the project. As a limitation in this study, the results are drawn only from the viewpoint of the development teams.

The content of this paper is as follows: the next section reviews the existing literature on DSD and agile DSD. This is continued by setting out the research settings. Section four presents the results, and a discussion of the results, and the paper concludes with final remarks and indications for future research.

2. Related literature

This section discusses DSD and distributed agile development.

DSD allows geographically independent software development for companies of all sizes and it is becoming a common practice in the modern software development industry [1, 3-5, 11]. Several factors have been contributing to the growing interest in distributed development, such as time-zone independent 24 hour development, reduced costs, access to well-educated labour and maturation of the technical infrastructure, just to name a few [4, 12-14]. Even though these factors inevitably provide tempting opportunities for software companies, DSD is troubled by the same problems as single-site efforts. Communication, requirements engineering, cost related problems, problems with quality and schedule are common issues in single-site development alone, and distribution makes these problems even more complex. [3, 4] Additionally, language and cultural differences create problems in DSD [12]. It has been also found that distributed development tasks take about 2.5 times longer to complete compared to co-located tasks due to communication and coordination related issues [9]. Komi-Sirviö and Tihinen [4] conducted an extensive study into charting the main problems encountered in DSD. Their results were categorized into nine specific problem areas, and the top three issues were *1) Development tools and environment, 2) Communications and contacts* and *3) Design knowledge*. Problems with tools and environment were the number one problem source within the large companies. The problems related to incompatible development tools and to the functionality of network connections. Medium sized and small organizations suffered mostly from design knowledge related issues, such as situations in which the software design and implementation take place in separate locations. While not being the main source of problems, 74% of the respondents identified communication as a problem in DSD. In general, the problem aspect of DSD seems to be so complex that a thorough understanding of it has not been reached. [4]

The research on distributed agile development has identified communication as one of the main issues to be taken into account [e.g. 5, 7, 10]. Agile software development involves highly volatile requirements which are managed through efficient verbal communication [8]. Effective communication is a crucial element in software production regardless of the development approach [8, 15, 16], and it can be considered even more important in agile software development due to its paramount role. In addition, the lack of informal communication results in lower awareness [5, 7] and poor coordination [17]. Awareness is defined as [18]: *"an understanding of the activities of others"*. Coordination helps individuals to view, plan and execute their actions in relation to their colleagues' actions towards a common goal. Awareness is a key concept affecting coordination in a distributed environment. [7]

Requirements engineering has also proven difficult in DSD [3, 4]. According to [4] requirements related issues were the major source of software errors, and communication makes this critical subject even more challenging [3-5]. Since in agile development the software requirements are documented on a very general level and communication can be considered cumbersome in distributed agile development, it is quite safe to argue that the volatility of agile requirements in conjunction with troublesome communication can create significant risks for distributed agile development efforts.

Several solutions for tackling the problems in distributed agile development have been proposed. These techniques include e.g. different support tools, such as instant messaging [9], videoconferencing [10] and whiteboard software [5]. Also more general level recommendations have been made. Layman et al. [5] have proposed guidelines presented in Table 1 for distributed agile development. They conducted a distributed case study using Extreme Programming [8]. The developers were located in the Czech Republic, while the management was situated in the USA.

Table 1. Recommendations by Layman et al. [5]

Proposed recommendations
a) *Define a person to play the role of the customer up front. This individual must be able to make conclusive decisions on project functionality and scope, must be readily accessible, and must have a vested interest in the project.*
b) *When the project management and development teams are separated, create a role within the XP team whose purpose is to work closely with both development and project management teams on a daily basis, preferably someone who speaks all the languages involved.*
c) *When face-to-face, synchronous communication is*

unfeasible, use an email listserv to increase the chance of a response and encourage prompt, useful, and conclusive responses to emails.
d) *Use globally-available project management tools to record and monitor the project status on a daily basis.*

These recommendations aim to create a communication-rich environment for projects unable to achieve it due to their distributed nature. Even though these recommendations have been developed as a result of agile GSD effort spreading over different time-zones, they could be applied also to efforts with a less distributed context.

To define the level of distribution we applied the model proposed by Prikladnicki et al. [2]. The model can be used to analyze the distribution through different scenarios in which the model's actors interact. The suggested model is set out in Figure 1.

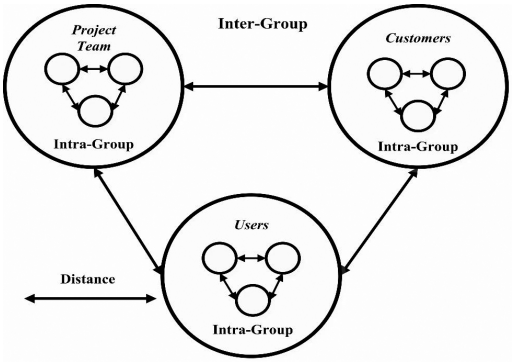

Figure 1. DSD Distribution Level [2]**.**

The model consists of the following actors: *Project Team, Customers* and *Users*. A *Project Team* includes everyone involved in the development including developers, business-oriented personnel, testers etc. *Users* represent people responsible for providing e.g. requirements, and *Customers* are the person(s) or organization that requested the project. The term *Intra-Group* defines the distance inside the different stakeholder groups, while *Inter-Group* refers to the distance between the different actors [2]. The stakeholder groups interact within different *scenarios* involving both intra- and inter-group distances [2]. According to the model, the level of distribution can range from relatively small, e.g. the *Project Team* and the *Customers* are located in adjacent buildings, to larger scenarios of global distribution [2].

3. Research design

This section presents the research method, research settings and data collection techniques. Also the levels of distribution in the case projects are described.

3.1 Research method, research settings and data collection

Our research covered two separate semi-industrial case studies [19] providing real software with real business value for real customers. The project team in *Case A* consisted of one experienced professional developer from the customer organization, and three experienced 5-6th year information processing science students working in a single shared environment. The target was to implement a mobile version of already existing software as an extension of that product. The customer was located in the same city and participated personally in the process during iteration planning and release phases.

In *Case B*, there were two development teams working on the same product and sharing a single customer. The project implemented a mobile front-end application that was integrated into the customer's main product. The front-end team was located in Oulu, Finland and the main product team in Helsinki, Finland was located approximately 600 kilometers away. The customer spent the first two weeks of the project with the Oulu team and for the rest of the effort he was located with the Helsinki team. Proxy customers were not utilized. The customer organization was located in Helsinki and *Case B* was not affected by time zone differences.

The Oulu team in *Case B* consisted of four to seven developers for the different iterations. This composition included an experienced developer and a Test Driven Development specialist.

The case projects were implemented following Mobile-D™, which is an incremental, iterative, agile inspired development methodology specially designed for mobile application development [20, 21]. Because of the limitations of this paper, Mobile-D™ is only briefly discussed. The data was collected using the following techniques: **onsite observations** in *Case A*, **final interviews** with the developers in *Case A* and the developers of the Oulu team in *Case B*. The interviews were semi-structured. A **personal research diary** and **email correspondence between the team and the customer** were utilized in both cases. In *Case B* the email correspondence between the customer and the Oulu team was examined. One team member from *Case B* was separately interviewed after the project using a semi-structured interview to cover issues not discussed in the final interview. This developer was selected to be interviewed since he was one of the most experienced developers and the one most easily available for the interview.

The customers were not interviewed which can be considered a limitation of the study.

3.2 Levels of distribution in the case projects

The case projects applied different levels of distribution. Following the distribution level model [2], the distribution in *Case A* appears in Figure 2. In both cases the *Users* and *Customers* were the same, which is often the case [2].

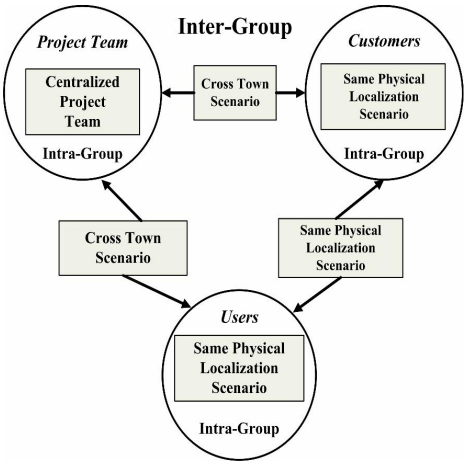

Figure 2. Distribution in Case A based on [2]**.**

Since the developers were sharing the same workspace, the *Project Team* was *centralized*, while the *Inter-Group* distribution between the team and the other stakeholders followed a *Cross Town Scenario*. There was only one customer, so the distribution within the *Customers/Users* was the *Same Physical Localization Scenario*. A similar analysis was conducted in *Case B*, and appears in Figure 3.

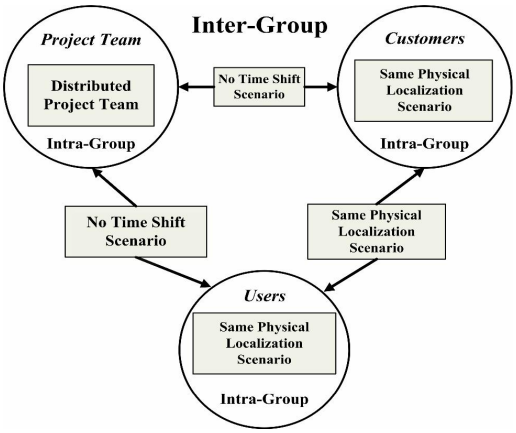

Figure 3. Distribution in Case B based on [2]**.**

In *Case B* the *Project Team* was distributed. The *Customers/Users* spent time with both of the teams, one team at a time. Therefore the *Inter-Group* distribution falls into a *No Time Shift Scenario* -category since the stakeholders were located in the same country without constant collaboration between all parties involved.

4. Results

In this section the results of the study are presented. We discuss our findings against the recommendations by Layman et al. [5].

Recommendation 1*: Define a person to play the role of the customer upfront. This individual must be able to make conclusive decisions on project functionality and scope, must be readily accessible, and must have a vested interest in the project.*

The study conducted by Layman et al. [5] indicates that the customer role is essential for effective requirements management in a distributed agile project, and active, consistent customer involvement cannot be overlooked.

The customer in *Case A* participated personally during the iteration planning and release phases. Our findings on this recommendation are drawn from *Case B*. During the *Case B* the customer was at first participating actively in the development with the Oulu team. He was available for continuous feedback for the first two weeks, afterwards he stayed with the Helsinki team. After his departure the communication was managed by daily emails and during the planning and release phases by telephone. The customer communication left a lot to be desired from the perspective of the developers. *When he [the customer] was present he was active, but when otherwise he should*

33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007)
0-7695-2977-1/07 $25.00 © 2007 IEEE

have been more active (On the customer communication. A Developer, Case B).

In addition to the lack of efficient customer communication, the communication of technical issues between the teams was not allowed. All communication between the teams was passed through the customer. Since the customer communication was not effective, it is not surprising that the Oulu team encountered significant problems during the development. The Oulu team, for example, did not receive any information about the communication protocol between the mobile front-end and the main application. This may indicate a lack of trust between the customer organization and the Oulu team. Prikladnicki et al. [22] identified trust as one of the difficulties in distributed development. Whether or not the lack of trust was the reason why the teams were not allowed to communicate directly, communication between separate teams should be allowed and maintained. This assumption can be supported by the finding made by Layman et al. [5]: *"The customer noted that cooperation and teamwork were the biggest success factors in this project"*. Thus active customer communication in conjunction with an environment supporting direct communication between the teams is essential for a successful result. The importance of peer-to-peer links is also mentioned in [23].

Recommendation 2*: When the project management and development teams are separated, create a role within the XP team whose purpose is to work closely with both development and project management teams on a daily basis, preferably someone who speaks all the languages involved.*

The project team in *Case A* was complemented with an experienced developer from the customer organization. This developer had a profound understanding of the software domain and he had been participating in the development of the desktop version of the same product. He had a strong working relationship with the customer so it seemed natural that he assumed the role of the *Development Manager* [5] – the role is described in recommendation two. However, he did not communicate with the customer on a daily basis. During the project the requirements analysis process was conducted almost invariably by the Development Manager and the customer. When the other team members were asked why they were not participating in the process, the answer was short: "*He [the Development Manager] knows what to do" (Developer, Case A)*. The team also mistakenly referred to the Development Manager as the "*Onsite customer"* even though he did not possess any power over the requirements definition or project scope. The passiveness of the team during the requirements analysis, along with the misunderstood role of the Development Manager, may indicate that the he was the main source of information for the rest of the team. The following comment from the final interview supports our view: "*If he [the Development Manager] hadn't participated in this project, we would have had to ask the customer a lot more questions and the requirements analysis would have been a lot more difficult" (A Developer, Case A)*. We argue that these comments indicate that having a member assuming the role of the Development Manager can create a false sense of confidence and security. The false sense of security is not a new issue in the field of agile development [24].

Another factor that needs to be taken into account if having a Development Manager is to emphasize that the requirements analysis process (for example) is not the responsibility of the Development Manager alone. We contend that allocating this task to a single person can reduce the possibilities of understanding the functionalities correctly. More people involving in requirements analysis process can reveal issues that can go unnoticed if the process is conducted only by the Development Manager and the customer. In addition, having a Development Manager may increase the chance of information distortion as, according to [25], information mutates and some of it gets lost if it passes through different people. In addition, similar viewpoints have been presented e.g. in [26]. In addition, it is also quite safe to argue that using intermediaries between the development and management sides is pointless if the customer is not able or willing to work actively with the Development Manager.

Recommendation 3*: When face-to-face, synchronous communication is infeasible, use an email listserv to increase the chance of a response and encourage prompt, useful, and conclusive responses to emails.*

During our research the daily iteration time communication between the developers and the customers was managed through email (*Case A)* and email and telephone (*Case B)*. These channels were specifically requested by the customers during both cases, despite the wide array of different existing communication channels (see section 2 for examples and [27]). Layman et al. [5] had positive experiences using asynchronous communication by utilizing mailing lists to share information. Their findings indicate that the customer was actively and constantly participating in the development, i.e. the customer was well-defined.

On the other hand, empirical evidence indicates that increasing reliance on so called lean, asynchronous communication channels can result in higher software defect rates [28]. During *Case A*, a total of 18 daily emails about the project were sent to the customer. All of these messages were brief status reports describing what was done during the day. Questions were not asked in these

reports. Thus it seems that the team was confident about what they were doing. During *Case B,* a total of 46 reports were sent. Three of these reports included questions for the customer, and the customer responded only to one issue. The average defect rate was 41.8%, while 62.6% of all defects were caused by inefficient customer communication [28]. These findings are in contradiction with the results by Layman et al. [5], indicating that this recommendation is strongly linked to the recommendation for a well-defined customer. Thus we conclude that if the customer is not able or willing to participate actively in the communication when needed, any communication mechanism becomes redundant.

Recommendation 4: ***Use globally-available project management tools to record and monitor the project status on a daily basis.***

Shared project management tools were not utilized in either of the cases.

In summary it seems that a well-defined customer (recommendation 1) is the key recommendation. If the customer is not able or willing to participate actively in the development, this makes the use of intermediaries between the development and customer sides, and use of mailing lists or any medium used in customer-developer communication, fruitless. In addition, we conclude that enabling and fostering direct inter-team communication deserves considerable attention. We propose this as an additional recommendation to the existing recommendations.

5. Conclusions

Distributed software development (DSD) is becoming a common practice in modern software development [5]. It both provides exciting opportunities [4, 12-14] and poses significant risks to the success of distributed projects [3-5]. Agile software development methodologies bring challenges to the field of DSD in the form of volatile requirements managed by informal communication. Several tools proposed in [e.g. 9, 10] have been used to tackle the problems of distributed development. Recent research has provided more general communication related recommendations focusing on distributed agile development [5]. The aim of this paper was to review these recommendations, and increase the knowledge about their usage based on the existing literature and empirical findings from two different small-scale distributed agile development efforts with different levels of distribution. Table 3 summarizes the recommendations made by Layman et al. [5] and illustrates how they were implemented in our research and are complemented with our empirical findings. We conclude by presenting our empirically based additional recommendation: *Enable and support direct communication between the developers.*

Table 3. Recommendations by Layman et al. [5] their utilization in this study and corresponding findings, including a new recommendation.

Recommendations [5]	**Utilized in this study**	**Findings**
1*: Define a person to play the role of the customer up front. This individual must be able to make conclusive decisions on project functionality and scope, must be readily accessible, and must have a vested interest in the project.*	During *Case A* the team had a single customer located in the same city. The customer was accessible during iteration planning and release. During *Case B* the customer spent the first two weeks constantly with the developers located in Oulu, Finland, afterwards he was not accessible. Both customers were in a situation to make decisions on functionality and scope.	Our findings from this area come primarily from *Case B*, concerning the team located in Oulu, Finland. The customer participated actively for the first two weeks; afterwards the communication was diminished radically. The customer replied only once to the email clarification requests sent by the team. This ineffective communication backfired with an average of 41.8% defect rate [28].
2: *When the project management and development teams are separated, create a role within the XP team whose purpose is to work closely with both development and project management teams on a daily basis, preferably someone who speaks all the languages involved.*	The Development Manager role [5] was utilized in *Case A.* The Development Manager was a member of the customer organization. The Development Manager was not communicating with the customer in daily basis. *Case B* did not have a Development Manager.	During our study the Development Manager seemed to be the main source of information for the developers in *Case A.* Personal customer communication e.g. requirements analysis was almost entirely managed by the Development Manager alone. It was admitted that without him the requirements analysis would have been a lot more difficult. We argue that having a Development Manager can create a false sense of confidence within

		the team. There is also a chance of information distortion e.g. [25, 26]. The whole team should be encouraged to participate e.g. in planning actions. However if the customer is not well-defined, the Development Manager becomes redundant. Thus this recommendation is linked to recommendation 1.
3: *When face-to-face, synchronous communication is infeasible, use an email listserv to increase the chance of a response and encourage prompt, useful, and conclusive responses to emails.*	While the customer was not personally available, the communication was managed by email (*Case A)*. Email, like a mailing list, is an asynchronous communication channel. In addition to email, also the telephone was used in customer communication (*Case B).*	Asynchronous communication can be used if all the parties involved in the development are committed to communicating actively. Empirical evidence has indicated that communication increasingly relying on asynchronous communication media can increase software defect rates accordingly [28]. This recommendation is highly linked to recommendation 1. If the customer is not able or willing to participate actively in the communication, the utilization of any communication mechanism becomes redundant.
4: *Use globally-available project management tools to record and monitor the project status on a daily basis.*	Not utilized in this study.	We have no findings from this area.
New recommendation: *Enable and support direct communication between the developers.*	The teams (*Case B)* were not allowed to communicate directly, but only through the customer who did not communicate actively with the Oulu team after the initial two weeks.	The teams should be able to communicate directly in order to achieve successful results. The lack of direct peer-to-peer communication can result in significant problems.

Even though the cases were not globally distributed as in [5], the presented recommendations can be viable also in a less distributed context, if they are fulfilled properly.

Based on the empirical data it seems that having a well-defined customer (recommendation 1) is the key to successful distributed agile development. Without proper customer collaboration, the other two investigated recommendations become redundant. Thus the customer relationship should be given extra effort in planning, managing and executing distributed agile projects. This relationship should also be maintained throughout the project.

As a limitation in this study, the results are drawn from the developers' points of view alone. Therefore, further research is required on the viewpoints of all the stakeholders for additional validation of all the existing recommendations, and possible discovery of new recommendations.

References

[1] J. Herbsleb and D. Moitra, "Global software development," *IEEE Software,* vol. 18, pp. 16-20, 2001.

[2] R. Prikladnicki, J. Audy and R. Evaristo. "Distributed software development: Toward an understanding of the relationship between project team, users and customers". In Proceedings of ICEIS, Angers, France 2003. pp. 417-423.

[3] D. Damian and D. Zowghi, "Requirements Engineering Challenges in Multi-site Software Development Organizations," *Requirements Engineering Journal,* vol. 8, pp. 149-160, 2003.

[4] S. Komi-Sirviö and M. Tihinen, "Lessons Learned by Participants of Distributed Software Development," *Knowledge and Process Management,* vol. 12, pp. 108-122, 2005.

[5] L. Layman, L. Williams, D. Damian and H. Bures, "Essential communication practices for Extreme Programming in a global software development team," *Information and Software Technology,* vol. Volume 48, pp. 781-794, 2006.

[6] C. Poole, "Distributed product development using

extreme programming," in XP2004, Garmisch-Partenkirchen, Germany. pp. 60-67.

[7] T. Schümmer and J. Schümmer, "Support for distributed teams in eXtreme programming," in ,1st ed.G. Succi and M. Marchesi, Eds. Addison-Wesley, 2001,

[8] K. Beck, *Extreme Programming Explained: Embrace Change.* Upper Saddle River, New Jersey: Addison-Wesley, 2000,

[9] J. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Transactions on Software Engineering,* vol. 29, pp. 481-494, 2003.

[10] M. Kircher, P. Jain, A. Corsaro and D. Levine, "Distributed eXtreme programming," in 2001, pp. 66-71.

[11] L. Taxén, "An integration centric approach for the coordination of distributed software development projects," *Information and Software Technology,* vol. 48, pp. 767-780, 2006.

[12] C. Ebert and P. De Neve, "Surviving Global Software Development," *IEEE Software,* vol. 18, pp. 62-69, 2001.

[13] R. Battin, R. Crocker, J. Kreidler and K. Subramanian, "Leveraging Resources in Global Software Development," *IEEE Software,* vol. 18, pp. 70-77, 2001.

[14] I. Gorton and S. Motwani, "Issues in co-operative software engineering using globally distributed teams," *Information and Software Technology,* vol. 38, pp. 647-655, 1996.

[15] R. B. Bostrom and B. D. Thomas, "Achieving excellence in communications: A key to developing complete, accurate and shared information requirements," in The Proceedings of the 20th Annual Computer Personnel on Research Conference 1983, Charlottesville, Virginia, United States pp. 1-13.

[16] A. Edstrom, "User Influence and the Success of MIS Projects: A Contingency Approach," *Human Relations,* vol. 30, pp. 580-607, 1997.

[17] J. Herbsleb and R. Grinter, "Splitting the organization and integrating the code: Conway's law Revisited ," in the proceedings of the 21st international conference on Software Engineering 1999, Los Angeles, CA, United States pp. 85-95.

[18] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *1992 ACM Conference on Computer-Supported Cooperative Work,* 1992, Toronto, Ontario, Canada. pp. 107-114.

[19] R. K. Yin, *Case Study Research Design and Methods.* Thousand Oaks, CA: Sage Publications, 1994,

[20] P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen and O. Salo, "Mobile-D: an agile approach for mobile application development," in OOPSLA 2004*,* Vancouver, Canada. pp. 174-175, 2004.

[21] T. Ihme and P. Abrahamsson, "Agile Architecting:The Use of Architectural Patterns in Mobile Java Applications," *IJAM,* 2005.

[22] R. Prikladnicki, J. Audy and R. Evaristo, "Global Software Development in Practice Lessons Learned," *Softw. Process Improve. Pract.,* vol. 8, pp. 267-281, 2003.

[23] M. Paasivaara and C. Lassenius. (2003, Collaboration practices in global inter-organizational software development projects. *Softw. Process Improve. Pract. 8(4),* pp. 183-199.

[24] T. Jokela and P. Abrahamsson. “Usability assessment of an extreme programming project: Close co-operation with the customer does not equal to good usability”. in PROFES 2004, Kansai Science City, Japan pp. 393-407.

[25] G. Melnik and F. Maurer, "Direct verbal communication as a catalyst of agile knowledge sharing," in AGILE 2004, Salt Lake City, Utah, United States pp. 21-31.

[26] M. Keil and E. Carmel, "Customer-Developer Links in Software Development," *Commun ACM,* vol. 38, pp. 33-44, 1995.

[27] E. Gottesdiener, *Requirements by Collaboration.* Addison-Wesley, 2002,

[28] M. Korkala, P. Abrahamsson and P. Kyllönen, "A case study on the impact of customer communication on defects in agile software development” in AGILE 2006, Minneapolis, Minnesota, United States. pp. 76-86.

PAPER IV

Combining Agile and Traditional: Customer Communication in Distributed Environment

Agility Across Time and Space – Implementing Agile Methods in Global Software Projects.
Eds. D. Šmite, N.B. Moe & P.J. Åkerfalk. Pp. 201–216.

PAPER V

Waste Identification as the Means for Improving Communication in Globally Distributed Agile Software Development

Journal of Systems and Software.
Volume 95, September 2014, pp. 122–140.

The Journal of Systems and Software 95 (2014) 122–140

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Waste identification as the means for improving communication in globally distributed agile software development

Mikko Korkala[a,*], Frank Maurer[b,1]

[a] VTT Technical Research Centre of Finland, Vuorimiehentie 3, 02150, Espoo, Finland
[b] Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4

ARTICLE INFO

Article history:
Received 28 November 2012
Received in revised form 26 March 2014
Accepted 26 March 2014
Available online 8 April 2014

Keywords:
Distributed agile software development
Lean software development
Communication

ABSTRACT

Agile approaches highly values communication between team members to improve software development processes, even though, communication in globally distributed agile teams can be difficult. Literature proposes solutions for mitigating the challenges encountered in these environments. These solutions range from general-level recommendations and practices to the use of communication tools. However, an approach covering the whole development process for identifying challenges, and improving communication in globally distributed agile development projects, is missing. In order to address this, we conducted a case study within a globally distributed agile software development project focused on using the concept of waste as a lens for identifying non-value producing communication elements. In order to achieve this, we constructed a waste identification approach through which we identified five communication wastes, and solutions to mitigate them. These wastes can help companies identify communication issues that are present in their development efforts, while the presented waste identification technique gives them a mechanism for waste identification and mitigation. This work contributes to the scientific community by increasing the knowledge about communication in globally distributed agile development efforts.

1. Introduction

Agile software development methods emerged during the late 1990s and early 2000s as a response to the industry's need for faster and more lightweight development approaches. One of the essential aspects of agile development methods is the emphasis on informal communication conducted preferably face-to-face (Beck, 2000). Informal communication has been defined by Kraut and Streeter (1995) as personal, peer-oriented and interactive. Further, Herbsleb and Grinter (1999) label informal communication as something that happens outside the official reporting structure of the project, and sometimes invoked without the knowledge of management. Informal communication enables correcting mistakes and filling in the required details fast (Herbsleb and Grinter, 1999). Physical proximity is essential for participants to engage in informal communication (Kraut and Streeter, 1995) and it is highly emphasized in agile literature. An agile development team should be located in a shared workspace and the customer (i.e. someone who will actively steer the project) should be on site to provide input and feedback (Beck, 2000). Close collaboration with the customers in agile software development has proven useful. For example, Ceschi et al. (2005) state that managing a customer relationship has proven easier in agile software development when comparing to traditional[2] approaches. 50% of the companies using traditional software development methods suffered from problems related to customer relationships. This is 40% higher than customer relationship issues encountered when using an agile development approach (Ceschi et al., 2005). Agile approaches are currently used in globally distributed environments that cross significant distances over time and space. This in turn makes physical, as well as temporal, proximity between the participants difficult to achieve and, hence, creates challenges for informal face-to-face communications (Nöteberg et al., 2003).

* Corresponding author. Tel.: +358 40 514 1042.
E-mail addresses: mikko.korkala@vtt.fi, mikko.korkala@gmail.com (M. Korkala), frank.maurer@ucalgary.ca (F. Maurer).
[1] Tel.: +1 403 220 3531.

[2] In this work, traditional development is synonymous with plan-driven development. Plan-driven software development is an engineering approach in which the software is developed following specific processes, commencing at the requirements gathering stage and ending with the final code (Boehm and Turner, 2003). Perhaps the most well known plan-driven method is the Waterfall method (Royce, 1970).

http://dx.doi.org/10.1016/j.jss.2014.03.080
0164-1212/

Effective communication is essential in globally distributed software development regardless of the development approach (Herbsleb et al., 2001; Carmel and Agarwal, 2001; Mockus and Herbsleb, 2001). In contrast to agile methods, traditional approaches rely on formal communication which codifies process and product knowledge into extensive documentation (Nerur et al., 2005). Kraut and Streeter (1995) describe formal communication as communication through structured meetings, writing and other relatively impersonal and non-interactive channels. A globally distributed context creates its own challenges that affect communication between distributed partners (e.g. Noll et al., 2010). According to the study made by Komi-Sirviö and Tihinen (2005), 74% of the problems of distributed development were related to communication. They found that the lack of communication or poor quality of it was often the root cause behind other problems identified in the study (Komi-Sirviö and Tihinen, 2005).

In order to address the communication challenges, different solutions have emerged. These proposals focus on general-level recommendations to establish an environment that fosters meaningful communication (e.g. Layman et al., 2006; Korkala et al., 2010), and the use of individual communication tools (e.g. Kircher et al., 2001; Danait, 2005). These recommendations provide solutions to particular problems such as organizational challenges (Korkala et al., 2010) and means to maintain interactive communication. In summary, there are no concrete approaches covering the entire development process to help companies analyze and improve communication in their globally distributed agile development projects. In our study, we use the concept of waste to illustrate communication specific bottlenecks. Waste is defined as *any human activity consuming resources but not providing value* (Womack and Jones, 1996) and this concept has also been applied in the context of Lean software development (e.g. Poppendieck and Poppendieck, 2007; Mandić et al., 2010). While these wastes are not limited to any particular aspect of software development, our study aims to identify communication specific wastes. Hence, the motivation of this study was to create an approach that covers the entire agile development process to identify communication specific waste and to classify what those wastes are.

We conducted a single case study within a North American software intensive company that was implementing a product in a globally distributed fashion. In the study, we applied the constructed waste identification approach and analyzed the communication between the involved stakeholders using the key concepts of Media Synchronicity Theory (MST) (Dennis et al., 2008). The communication wastes were extracted from the data and we propose actions for mitigating the effects of the identified wastes.

The contribution of this paper is twofold. First, we propose a waste identification process through which the non-value producing communication elements can be identified. As a second contribution, we identified five types of waste related to communication; ***lack of involvement***, ***lack of shared understanding***, ***outdated information***, ***restricted access to information*** and finally ***scattered information***. This study concludes that the proposed waste identification process can point out non-value producing elements from communication, after which, measures to mitigate them can be identified.

The rest of the paper is organized as follows: Section 2 discusses background literature relevant to this study in order to provide the reader an understanding of communication in globally distributed environments. Section 2 also discusses the concept of waste in the context of software development and explains the communication theory through which communication was analyzed. Section 3 presents how the study was conducted and introduces the waste identification approach. Section 4 presents the findings of this study, and generalized descriptions of the wastes alongside proposals for mitigating them. Threats to validity are also discussed in this section. In Section 5 the findings are discussed along with future research opportunities. Finally, the conclusions are drawn with more detailed description of how to apply the presented waste identification approach.

2. Related literature

In this section, background literature relevant to the study is discussed. First, we discuss communication in the context of globally distributed software development from the perspectives of both traditional and agile software development, after which we introduce Media Synchronicity Theory. Finally, we address the concept of waste. These elements provide the theoretical framework for our study.

2.1. Communication in globally distributed development environments

Globally distributed software development (GSD) is a common approach in software engineering (Damian and Moitra, 2006) and several factors have contributed to the growth of this phenomenon. Some of the most commonly cited benefits include time-zone independent "follow the sun" development, access to well-educated labour, maturation of the technical infrastructure and reduced costs due to different salary structures based on geographical regions (Komi-Sirviö and Tihinen, 2005; Ebert and De Neve, 2001; Gorton and Motwani, 1996; Battin et al., 2001). In order to achieve these benefits, communication between the distributed parties must be effective. However, there are several challenges that hinder communication in distributed contexts.

Noll et al. (2010) have identified geographical, cultural and temporal distances as the key barriers for communication and collaboration in globally distributed environments. Holmström et al. (2006) state that it is the combination of these distances that makes globally distributed development complex. Table 1 presents challenges found in these categories both from the general GSD literature (not focusing on any particular development approach) and those found in agile GSD literature.

Several different approaches to address these challenges have been suggested. The reduced opportunities for synchronous and face-to-face communication from both geographical and temporal viewpoints can be mitigated by using interactive communication tools, such as videoconferencing (Kircher et al., 2001; Sureshchandra and Shrinivasavadhani, 2008), whiteboard software (Layman et al., 2006), web conferencing and Instant Messaging tools (Danait, 2005). Asynchronous tools can also be used. According to Damian and Zowghi (2003), email is the dominant asynchronous media due to its important role as a means of exchanging documents across temporal distances. The use of asynchronous media can, however, create challenges. Email messages can be forgotten or lost which leads to unresolved issues, and the time when a response to an email message will arrive is unsure (Damian and Zowghi, 2003). Carmel and Agarwal (2001) further state that asynchronous media often delays problem resolution and makes it more complicated. As an example, even simple issues can take days of email discussion before resolving (Carmel and Agarwal, 2001). In order to reduce communication overhead, strict communication policies are promoted if asynchronous communication tools are used, such as, emails need to be replied within 12 business hours as suggested in Vax and Michaud (2008).

Communication tools themselves can create challenges. Sound quality of teleconferencing tools may be poor. This can create misunderstanding and communication overhead can occur when messages need to be repeated several times (Williams and Stout, 2008). Additionally, significant amounts of time may be needed

Table 1
Challenges of GSD and agile GSD from the perspectives of temporal, geographical and cultural distances.

Challenge area	Findings
Temporal distance	**General GSD literature** Opportunities for synchronous communication are reduced (Ågerfalk and Fitzgerald, 2006). Communication needs to take place on unconventional times due to the lack of overlapping working hours and leads to overtime work. This is consuming and leads to communication overhead (Holmstrom et al., 2006; Conchúir et al., 2009; Sarker and Sahay, 2004). Possible unavailability of remote colleagues when help is needed can lead to delays. Asynchronous communication media used over temporal distance increases response times (Ågerfalk, 2004). **Agile GSD literature** Using interactive media for efficient communication can be very difficult due to temporal distance (Korkala et al., 2010).
Geographical distance	**General GSD literature** Face-to-face meetings are difficult to arrange and informal communication is lacking (Ågerfalk and Fitzgerald, 2006). This inhibits idea sharing (Conchúir et al., 2009). **Agile GSD literature** Explicit findings not found. However, the identified challenges can be seen valid in globally agile development as well.
Cultural distance	**General GSD literature** Misunderstandings in communication stemming from cultural differences (Holmstrom et al., 2006; Ågerfalk and Fitzgerald, 2006; Conchúir et al., 2009). **Agile GSD literature** Misunderstandings in communication stemming from cultural differences (Summers, 2008). Cultural differences may result into situations in which e.g. disagreements are not willingly expressed and negative issues are shared reluctantly (Lee and Yong, 2010; Drummond and Francis, 2008). Language barriers can significantly hinder communication (Layman et al., 2006; Uy and Ioannou, 2008; Kajko-Mattsson et al., 2010). Different work styles cause communication problems (Sutherland et al., 2007).

to resolve technical issues with videoconferencing (Williams and Stout, 2008). Poor technical communication infrastructure can create such challenges that meetings may even need to be rescheduled due to poor sound/video quality (Therrien, 2008). Other times, it is not even possible to use videoconferencing tools (Paasivaara et al., 2008; Herbsleb and Moitra, 2001). Considering the consuming nature of unconventional overlapping work hours that enable synchronous communication, it is recommended that work should be kept sustainable (Therrien, 2008).

In order to overcome cultural hurdles there are several recommendations available. For example, experienced domain experts should communicate with distributed teams daily. This should mitigate communication risks emerging from cultural differences by keeping the potential problems transparent (Summers, 2008). In addition, creation of specific roles has been proposed as an approach for mitigating communication issues stemming from cultural differences. Layman et al. (2006) proposed the creation of a role responsible for close collaboration between the developers and the management, in the event that they are separated. Korkala and Abrahamsson (2007) studied the propositions of Layman et al. (2006) and concluded that they are worthwhile to consider in distributed settings. Furthermore, they emphasize the importance of direct communication links between the participants. It should be noted however, that the distribution in the case project was not on a global scale (Korkala and Abrahamsson, 2007).

Bureaucratic organization can create barriers for communication in agile GSD projects. Bureaucratic organizations are characterized as hierarchical, procedural, regulated, established, structured, cautious and power-oriented (Wallach, 1983), and they have been identified as a difficult environment for agile development projects (Berger, 2007). Korkala et al. (2010) found supporting evidence for this claim in their study focusing on analysing communication in a globally distributed agile development project that had separate customer and vendor organizations involved. They found that bureaucratic organizational culture created significant challenges for communication. One of the challenges was deliberate information hiding, i.e. the customer organization did not grant the vendor access to important information such as their codebase. Further, the customer organization did not provide adequate information about the requirements for the vendor. The lack of domain knowledge at the vendor organization made their situation even more challenging. In order to respond to these findings, the study concludes with a set of guidelines aiming at creating an environment that supports meaningful communication (Korkala et al., 2010).

The literature study made for the purposes of this work shows that communication challenges of GSD are many, and exist regardless of the development approach. It also shows that the mechanisms for mitigating these challenges are merely general-level recommendations and encouragements for using individual communication tools. This finding, combined with the pivotal role of communication in agile development, further supports the purpose of this study to provide a concrete approach for companies to improve communication in their agile GSD projects.

2.2. Media Synchronicity Theory

Communication theory and the processes through which communication was analyzed during the study are presented in this section. An overview provides understanding of the main concepts that created the theoretical lenses of the study from the perspective of communication.

Media Synchronicity Theory (MST) is an extension of Media Richness Theory (MRT) (Daft and Lengel, 1986; Daft et al., 1987) and aims to predict the performance of communication and examine communication media capabilities in various contexts of use (Dennis et al., 2008). MST defines two separate communication processes, *conveyance* and *convergence*. In our study, we use these processes to categorize and analyze the use of different communication media in different phases of the case project.

Conveyance is related to transmission of new information that enables the receiver to create and revise a mental model of the information. In order to establish this understanding, as much relevant information as necessary is required. If information is insufficient (i.e. conveyance is defective), incorrect conclusions will be reached. ***Convergence*** process aims towards a shared

Table 2
The ability of different communication media to support synchronicity as defined in Dennis et al. (2008).

Communication medium	Ability to support synchronicity
Face-to-face	High
Video conference	High
Teleconference	Medium
Synchronous instant messaging	Medium
Email and asynchronous electronic communication	Low
Voice mail	Low
Fax	Low
Documents	Low

understanding on the meaning of information, and the participants need to mutually agree that the mutual understanding is achieved or that it cannot be achieved ("agree to disagree"). Convergence can require less information processing compared to conveyance if it focuses on a smaller set of information, for example a particular detail, than what was conveyed in the first place. However, if large differences in individual understandings exist convergence may require as much cognitive processing than conveyance. Defective convergence prevents individuals moving forward to other activities since they lack a shared understanding (Dennis et al., 2008).

Synchronicity is defined in MST as "*the ability to support individuals working together at the same time with a shared pattern of coordinated behaviour*". Further, media synchronicity is defined as "*the extent to which the capabilities of a communication medium enable individuals to achieve synchronicity*" (Dennis et al., 2008). Synchronicity is not always easy to achieve. Dennis et al. (2008) postulate that synchronous communication is necessary to synchronicity, but it is not necessarily sufficient considering participants can lack a shared focus during communication. The participants can for example, read their email during a meeting.

Using media with lower synchronicity should increase performance for conveyance processes (i.e. sharing new complex information), while convergence processes (agreeing on details, for example) benefit from using media with higher synchronicity (Dennis et al., 2008). In addition, a convergence process typically requires rapid transmission of small quantities of pre-processed information back and forth between the participants (Dennis et al., 2008). Communication media vary in their properties to support synchronicity. Table 2 is a summary from Dennis et al. (2008) depicting media abilities to support synchronicity.

According to Dennis et al. (2008), no single medium is inherently better than another and successful completion of tasks requires both conveyance and convergence processes. Therefore, it is proposed to use different media either simultaneously or in succession. Dennis et al. (2008) further suggest that the situation in which a medium is used affects its suitability for particular communication situations. The communication processes, the individuals engaged in communication and the social context in which the communication takes place all affect the medium's suitability for the given situation.

Dennis et al. (2008) also discuss *appropriation factors* that influence how people use different media. These factors further propose that when communicating participants' familiarity with each other increases, the need for media that supports high synchronicity is reduced (Dennis et al., 2008). Thus, the need to use different communication media is not constant, but evolves in time. In addition, media that fit the users' needs are more likely to be appropriated and used faithfully than those not meeting these needs (Dennis et al., 2008).

Considering the validity of MST, Niinimäki et al. (2010) studied twelve distributed projects and found evidence that MST can be used for selecting communication tools for projects operating in globally distributed settings. The study concludes that even though the tool use and decisions made on media choice were not always following suggestions of MST, the ideas presented in the theory are applicable and useful in globally distributed development projects. The results of a laboratory experiment on the theoretical underpinnings of MST are reported in Dennis et al. (1998). The study focused on the impacts of different media on effectiveness of both conveyance and convergence. The study provided preliminary support for MST. DeLuca and Valacich (2006) studied MST through seven hypotheses and found support for MST. Further, Dennis et al. (2008) provide a list of studies whose findings they explain via MST. Originally these studies applied MRT, whereas MST was able to answer to unexplained results of these works.

2.3. The concept of waste

The origins of waste can be traced back to the Japanese automobile industry of the 1950s and more specifically to Toyota Production System (TPS) (Ohno, 1988). The literature discusses two kinds of waste: *Type 1 waste* involves non-value adding activities that cannot be removed or mitigated from the current operating environment while *Type 2 waste* is a non-value adding activity that can be removed or mitigated (Womack and Jones, 1996). While the manufacturing industry has its own defined wastes, Poppendieck and Poppendieck (2007) have mapped these wastes to software development activities. These wastes, complemented with those identified by Mandić et al. (2010) and their descriptions, are discussed in Table 3.

A case study reported by Ikonen et al. (2010) revealed that the wastes presented in Poppendieck and Poppendieck (2003) are relevant to software engineering. It was concluded that software development projects can be successful even though waste exists and waste seems to be something that cannot be avoided in software development projects (Ikonen, 2010). There seems to be certain challenges in identifying waste in software development. Usually, in production and manufacturing the underlying nature of waste is visible and generally well understood (Hicks, 2007). In his work, Hicks (2007) argues that in the context of information management, the situation is the opposite. Ikonen et al. (2010) share a similar understanding around software development.

3. Research design

In this section, we describe how our study was conducted. The case study and its context are discussed along with the data collection and analysis techniques. In addition, we present the waste identification process.

Even though research in software engineering has a result-oriented, pragmatic view on research methodologies (Seaman, 2002), the philosophical perspective on this study can be seen as an *interpretative* case study. Orlikowski and Baroudi (1991) claim that an interpretive study attempts to understand a particular phenomenon (in this case, communication), through how the participants interpret their context. Hence these studies try to increase the understanding of this phenomenon and use this knowledge to inform other settings.

Case studies typically aim at answering "how" and "why" questions (Benbasat et al., 1987). In this study, we formulated three research questions. The main research question is:

- ***RQ:*** *How can waste identification improve communication in globally distributed agile software development?*

Table 3
The wastes of software development and their descriptions based on Poppendieck and Poppendieck (2007) and Mandić et al. (2010).

Waste	Description
Wastes and their descriptions identified by Poppendieck and Poppendieck (2007)	
Partially done work	Something that is not completed. E.g. untested code, undocumented or not maintained business decisions.
Extra features	Something that is not really needed.
Relearning	E.g. forgetting decisions, re-trying solutions already tried and the inability to utilize the knowledge of other people.
Handoffs	Passing work to someone else, getting work from someone else.
Task switching	How many other tasks the people need to do. Switching between tasks is distracting. For example, often switching between three or four smaller tasks requires more time to re-orientate to those tasks than to work on those. The time required to re-orientate to a task is waste.
Delays	Waiting for something.
Defects	Errors in program code.
Wastes and their descriptions identified by Mandić et al. (2010)	
Avoiding decision-making	This waste is about avoiding decision-making altogether and it should not be confused with "*Defer commitment*" practice of lean software development (Poppendieck and Poppendieck, 2007). The reasons behind this waste can vary from organizational issues, such as lack of empowerment, to individual characteristics.
Limited access to information	This waste is related to the existence of information; it might not exist. Limited access to relevant information may result in harmful decisions.
Noise or information distortion	Further divided to the dimensions of time and space. *Time related distortion* occurs when information is not recorded, not updated or is forgotten. *Space related distortion* occurs because actors are distributed across different levels or units of an organization and representing different contexts and sub-contexts. Space related information distortion is a phenomenon described by Melnik and Maurer in (2004).
Uncertainty	A variable or choice can have multiple values or options. These increased values or options increase the level of uncertainty.

This main research question is further elaborated by the following sub-questions:

- ***SQ1:*** *How can communication waste be identified in globally distributed agile software development?*
- ***SQ2:*** *What wastes can be found in communication in globally distributed agile software development?*

In this study, we identify waste that is specific to communication in globally distributed agile software development. We do not classify previously identified waste as communication specific in spite that they can be linked to communication in agile GSD. Similarly, we exclude well-known issues in GSD from being classified as communication waste.

The unit of analysis in this study was a single globally distributed agile software development project within a medium-sized North American software intensive company that implements interactive products for education and business. This site was selected based on the following criteria:

1. The case organization had operated using agile approaches for a period of nearly three years at the time of the study and had extensive experience working in a globally distributed context.
2. The case organization provided extensive access to individuals at multiple levels within the organization. These individuals were able to describe the situation from different viewpoints.

The project to we studied was suggested by the case company's management since it was working clearly in a globally distributed fashion with different tasks allocated to various sites across continents. In addition, management and the project personnel provided the authors extensive access to relevant data that was gathered using several different data collection techniques. The discussions with management and the project manager prior to the study revealed that they were, in general, satisfied with the project's progress. However, they identified the increased demands for communication with the offshore development organization as a challenge.

The project developed an improved version of an existing product that had been implemented by the case organization over the last 10 years. The software was divided into high-level units of functionality labelled as themes by the case organization. These themes contained different sets of functions that the product was supposed to provide. The themes were allocated to three different sites, remaining as independent as possible from the themes that were allocated to other locations, with the idea to minimize dependencies. Two of the sites were located in North America and were part of the case company. One site was located in India. This site was an independent contractor organization. Fig. 1 depicts the project organization along with temporal distances between the sites.

Initial Product Backlog was created before the actual programming work began. This process is described later in the paper. The programming work was not initiated at the same time in all sites. NorthAmerica 2 began development in September 2010. The programming work began at NorthAmerica 1 in June 2011 and in May 2011 at India. The reason why NorthAmerica 2 started earlier than the others was that the largest themes from the Product Backlog were assigned to them while the rest of the personnel allocated to the project at other sites were fully committed to other efforts. India had been collaborating with the parent organization for the last two years. NorthAmerica 1 was leading the development and contained the personnel responsible for steering the development along with the Product Owner. Before the implementation began at NorthAmerica 1, their leading role focused on steering the development at other sites.

3.1. Data collection

The application of several data collection methods such as archives, observations, interviews and questionnaires is often used in case studies in order to increase their validity (Eisenhardt, 1989; Yin, 1994; Stake, 1995). In our study, the data was collected using observations, informal discussions, documents provided by the case organization, and semi-structured interviews with open-ended questions. This data was collected at different phases of the project. The timeline of the study is presented in Fig. 2. The study had two main phases. The first phase focused on obtaining a solid overview of the case project while the latter phase focused on acquiring detailed information from the participants in the form of interviews. The actions described in Fig. 2 in some instances overlapped despite being presented in a linear fashion for the sake of clarity. Data collection and analysis were conducted throughout the project. The one-month gap between the phases is due to a vacation. The timeline presents the research activities after site and case project selection.

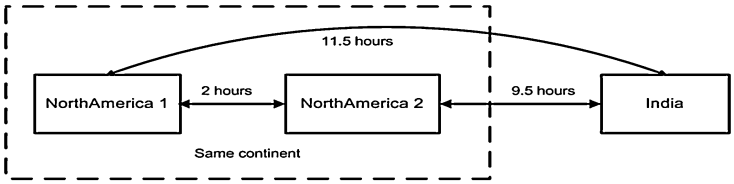

Fig. 1. The project organization along with temporal distances between the sites.

When the study was initiated all sites were involved in the project. NorthAmerica 1 and India were in the process of clarifying their individual requirements at the point when the study began. NorthAmerica 2 was implementing the product requirements assigned to them.

In order to ensure that the research data would be obtained from all necessary viewpoints, the key stakeholders of the project were identified together with the project manager, and the case company management. Table 4 presents the interviewed project stakeholders. In order to unambiguously identify stakeholders, their corresponding codes are preceded by the site name when there is a possibility of ambiguity in stakeholder location.

The study participants at NorthAmerica 1, excluding the developers, were team members who in addition to their work responsibilities worked as *Product Owner intermediates* for the teams relaying information from the Product Owner to the teams and vice versa. In addition, these senior members supported the Product Owner in decision-making on features to be implemented and provided direct feedback to the other sites.

The initial understanding of the project and the product was obtained during a total of 11 observation sessions taking place either onsite at NorthAmerica 1 or via telephone and screen sharing software. Field notes were taken during these sessions and a research diary was constantly updated during the study. In addition to observations, documentation about the requirements gathering and analysis process, as well as informal discussions and emails served as the means of providing information. If information was ambiguous, clarifying questions were asked until mutual understanding was achieved. In these observations, communication between NorthAmerica 1 and NorthAmerica 2 and between NorthAmerica 1 and India was studied.

During the first stage of data collection, one 70-minute interview focusing on the use of documentation and tools for sharing and storing it was conducted face-to-face at NorthAmerica 1. This interview involved all NorthAmerica 1 participants excluding the developers.

Data collection in research phase 2 was conducted via interviews. A total of 12 semi-structured interviews with open-ended questions and probes were conducted. Each interview lasted from 60 to 90 minutes and all participants were interviewed once. The interviews at NorthAmerica 1 were conducted face-to-face, while Skype was used in interviews involving NorthAmerica 2 and India. Due to company policies at India and to help with language related issues India.ProjectManager participated in all the interviews conducted with India. All the interviews were recorded and transcribed verbatim. In addition, field notes were taken and the research diary was updated.

3.2. Waste identification process

The project's development process contained two main phases labelled *pre-development* and *development*. As the names suggest, the pre-development phase included work conducted before the implementation begun. In this phase we focused on communication during the definition of the initial requirements and the initial Product Backlog. The development phase followed a Scrum approach (Schwaber and Beedle, 2002; Schwaber, 2004) with fixed-length three week synchronized Sprints (i.e. all sites began and ended at the same time). In this phase, we focused on analysing communication during identified development steps. The analysis of documentation as a communication tool during the project was included in the waste identification process. Fig. 3 describes the waste identification process applied in this study.

The use of different communication channels used by key stakeholders during different process steps is analyzed and related communication wastes are extracted in this process. The improvement actions responding to these challenges are determined and applied in the development process.

In our study, the **input** for each analysis step was a set of questions used to obtain a holistic view of communication within each phase. The questions focused on the steps belonging to Pre-Development and Development phases were based on the following main topics: *(1) who was involved in communication during the step, (2) what media were used during the step, (3) what kind of information was discussed during the step, (4) what were the benefits*

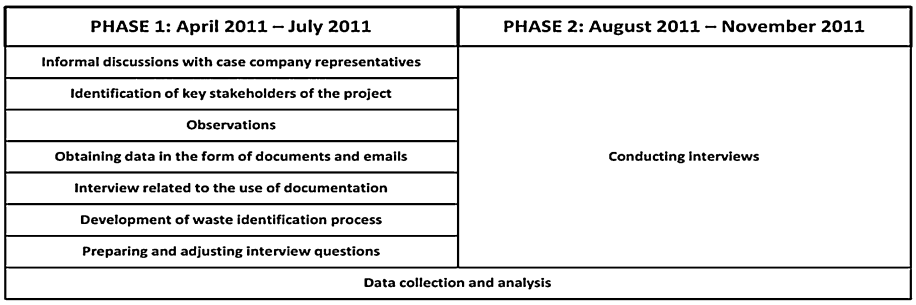

Fig. 2. The timeline of the study along with research activities.

Table 4
The interviewed stakeholders at different sites along with the description of their main responsibilities and their code in this study.

Role	Description of main responsibilities	Code
NorthAmerica 1		
Product Owner	Responsible for providing an overall strategy for the product. Identification and prioritization of new features and steering of development.	ProductOwner
Project Manager	Project planning and defect monitoring. Responsible for timely delivery of the product that meets both quality and financial requirements.	ProjectManager
Personnel Manager	Responsible for coordinating people based on the demands of technical decisions, for example, supporting the growth of peoples' technical competences and well-being.	PersonnelManager
User Experience Specialist	Creating user interface drafts, conducting usability testing and user research. Supporting development teams in usability aspects. Discussing user interface & usability topics with the ProductOwner and the development teams.	UserExperience
Overall Technical Lead	Responsible for technical aspects of the whole project. Responsible for technical architecture of the product and communicating technical details of the requirements to all teams.	MainTechLead
Two Developers	Responsible for developing the software at NorthAmerica 1.	NorthAmerica 1.Dev1 and NorthAmerica 1.Dev2
NorthAmerica 2		
Technical Lead	Responsible for technical aspects of the product at NorthAmerica 2.	NorthAmerica 2.TechLead
Developer and Tester	Developer was responsible for developing the software at NorthAmerica 2 while Tester was responsible for creating automated test cases for the product and worked as a coordinator between the testing services team and other teams working with the project.	NorthAmerica 2.Dev and NorthAmerica 2.Tester
India		
Project Manager	Responsible for delivering the decided contents of each Sprint at India.	India.ProjectManager
Quality Assurance Engineer	Responsible for the quality of the software developed at India.	India.QualityAssurance
Two developers	Responsible for developing the software at India.	India.Dev1 and India.Dev2

of the media when used during the step, and *(5) what were the wastes in communication during the step*.

Questions related to Documentation involved the following topics: *(1) what documents were produced during the development*, *(2) who was involved in the creation and updating of documents*, *(3) what were the benefits and wastes of using documents as a communication tool*, and *(4) how were the documents stored and what issues did this cause*. The **output** from each step was the general overview of communication during the steps (who participated, what was discussed and using what media), the benefits of communication media used

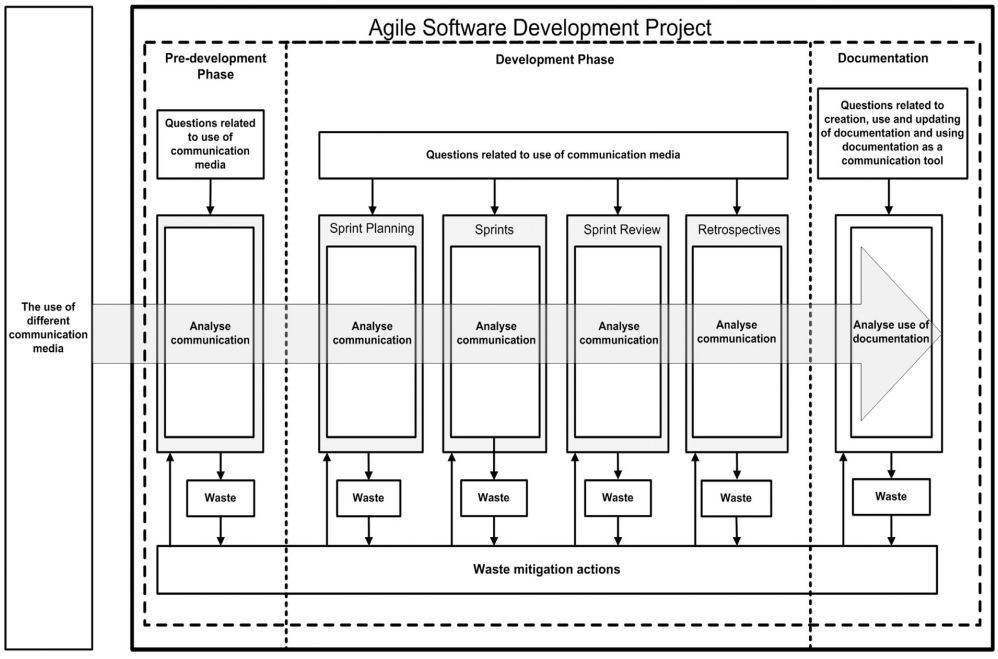

Fig. 3. The waste identification process applied in the study.

in them, and the non-value producing elements of communication from each step. Similar output was acquired from the perspective of documentation. Waste mitigation actions were defined in the *waste mitigation actions* process step.

This approach is derived from a framework described in Pikkarainen et al. (2008). Through their framework, Pikkarainen et al. (2008) analyzed whether the use of agile practices improved or hindered communication between the stakeholders of the organization. While communication hurdles are discussed in their approach, Pikkarainen et al. (2008) do not discuss communication waste. Further, their approach considers the whole organization including management while our approach focuses entirely on the development project level.

3.3. Data analysis

After the interviews, the transcriptions were first read and relevant topics were coded. Prior to the study a set of initial codes ("seed categories" (Miles and Huberman, 1994)) were created. As in Thematic Analysis (Braun and Clarke, 2006), additional categories were added when seen relevant. During the analysis a data reduction process (Miles and Huberman, 1994) was followed in order to focus on the essential data. The notes from informal discussions and observations were analyzed after the sessions from which they were collected. After the coding the tagged questions and resulting answers were grouped into a separate table forming a data display containing the compressed assembly of information, as proposed by Miles and Huberman (1994). The information was complemented with data available from field notes, observations and discussions. Wastes were extracted from the data. We also analyzed if the communication in each phase aimed towards conveyance or convergence and if the used media was capable of supporting these processes. This enabled us to discuss our findings from the perspective of MST.

Our data analysis has similarities to Grounded Theory. Cruzes and Dybå (2011) summarize Grounded Theory from Glaser and Strauss (1967) and Corbin and Strauss (2008) as a research approach that describes methods for qualitative sampling, data collection and data analysis. Grounded Theory includes simultaneous data collection and analysis phases (which was also applied in our study). However, Grounded Theory aims towards a generation of new theories, which was not the aim of our work. Further, in Grounded Theory the codes through which the data are analyzed are not pre-defined. Instead, they emerge from the data. Hence, the similarities between Grounded Theory and this study are limited to overlapping data collection and analysis.

4. Findings

This section discusses the findings of the study categorized by identified development process phases including the use of documentation. The identified communication specific wastes are summarized in each section. Further, we provide generalized descriptions of the identified waste as well as means to mitigate them in this section. In addition, the findings are discussed from the perspective of MST and threats to validity are addressed.

4.1. Pre-Development

Prior to entering the actual development phase, the ProductOwner collected a set of product requirements from real end-users, through the feedback system built into the older version of the product as well as by analysing competitors and market trends in the domain. In addition, internationalization needs affected the requirements. The collected requirements were further divided into 12 different themes that depicted functionalities at a high level.

Table 5
The communication media used during Pre-Development phase.

Communication within NorthAmerica 1	Communication between NorthAmerica 1 and NorthAmerica 2	Communication between NorthAmerica 1 and India
Backlog creation was conducted in face-to-face sessions at NorthAmerica 1.	NorthAmerica 2 did not participate in the backlog creation process with NorthAmerica 1. However, they provided input for themes allocated to them via telephone and screen sharing. Also email was used for exchanging information.	India did not participate in the backlog creation process. They did not communicate with NorthAmerica 1 before the development began.

These themes were split up between the three sites. Each site was given the responsibility to refine the themes to user stories with support from NorthAmerica 1. Table 5 depicts the usage of different communication media during Pre-Development phase.

4.1.1. Findings from NorthAmerica 1

The initial Product Backlog was created in face-to-face brainstorming sessions at NorthAmerica 1. All the stakeholders from NorthAmerica 1 participated in these sessions, excluding NorthAmerica 1.MainTechLead and UserExperience who were not working for the project at the time when these sessions took place. During the backlog creation process the PersonnelManager encountered problems in understanding one of the new product features. This particular feature was described in the initial requirements list in a very general way (PersonnelManager): "*It was a very high-level statement like containers will accept or reject objects*". This caused the PersonnelManager to seek additional information in order to provide accurate information to the ProductOwner about the functionality:

> "I needed a little bit of coaching to help me understand what it really was. I couldn't see how somebody would really use that unless it was simple to use and set up. You really have to understand the details in order to give a good detailed analysis of how something like that could be built." (PersonnelManager)

Also the ProductOwner saw similar challenges in the high level requirements: "*Our requirements documents are sufficiently high-level that it leaves a lot open to interpretation, which has its benefits but also has its drawbacks in that it's difficult to define from the requirements documents specifically what is the exact requirement. At that point it's not really broken down to the story level, sometimes it's just a major theme.*"

The extra effort needed to acquire additional information about vaguely communicated requirements can be seen as wasteful. However, these findings are stemming from the inherent characteristics of agile development since deliberately vague requirements are only clarified as needed. Therefore, these findings are not treated as waste in the context of this study.

4.1.2. Findings from NorthAmerica 2

NorthAmerica 2 did not participate in brainstorming sessions but they provided input for certain themes identified by NorthAmerica 1. These conversations took place via telephone and screen sharing software. The information shared in these conversations was further converged (i.e. more details were provided) via email and telephone. The issues NorthAmerica 2 experienced in this phase emerged from the requirements documentation.

Table 6
The identified communication related waste from Pre-development phase.

Identified waste	Description in the context of the phase
Lack of involvement	The lack of involvement from India to the initial backlog creation process resulted in insufficient understanding of real end user needs. This further caused *extra features*.

> "Some information in the documents that we received wasn't complete. There wasn't enough information in order for us to do the work. So not being co-located, it wasn't as easy as just walking over to someone, asking what did you mean by this." (NorthAmerica 2.TechLead)

The requirements were documented on a high level following agile approaches, so the incompleteness of the information in this case is something that can be expected. In a distributed context, difficulties of engaging in informal discussions over geographical distance are a known issue. Filling in the missing details (i.e. converging on the details) was more laborious, but it did not cause any particular issues.

> "We tried to spend a little bit more time investigating, and then writing up emails or getting telephone calls to try to get that information. It was just getting the information in order to do the proper research. When it was missing it was just harder to get." (NorthAmerica 2.TechLead)

These findings indicate similar issues with conveyance as in NorthAmerica 1; the information provided for NorthAmerica 2 was not conveyed efficiently. However, NorthAmerica 2 was able to compensate the lack of information by converging on the missing details even though it was more laborious due to geographical distance.

4.1.3. Findings from India

India did not participate in the backlog creation process and had no communication with NorthAmerica 1 before they started developing the features assigned to them. India did not report any specific issues that had emerged from documenting requirements at a high level. However, they experienced one particular issue during the development that stemmed from not participating in the backlog creation process.

> "Sometimes it happens that we suggest some features which may be very good for a customer, and sometimes it is accepted. And sometimes it happens that they (NorthAmerica 1) say that the users basically don't require or don't like this feature. So, this basically is a gap because we don't have a clear picture of what the end-user wants, what kind of features the user basically prefers." (India.Developer)

Therefore, there were *extra features* at India resulting from not participating in the initial backlog creation process. In this case, the waste behind the issues was ***lack of involvement*** stemming from India's absence during backlog creation process. A summary of the communication related waste from this phase is described in Table 6.

The requirements were documented following agile approaches, which, from the perspective of MST, rely on convergence instead of conveyance. At NorthAmerica 1, the initial backlog creation process was conducted using media with high support for convergence. This approach was not free from problems. Considering the extra effort for understanding the requirements, this could have been mitigated by more efficient conveyance, e.g. more detailed documentation, which on the other hand is counterintuitive with the propositions of agile development.

Table 7
The communication media used during Sprint Planning phase.

Communication within NorthAmerica 1	Communication between NorthAmerica 1 and NorthAmerica 2	Communication between NorthAmerica 1 and India
Face-to-face meetings	The decisions considering the Sprint contents were agreed via telephone, supported by screen sharing software.	The decisions considering the Sprint contents were agreed via telephone, supported by screen sharing software.

Defective conveyance due to vaguely defined requirements was encountered at NorthAmerica 2 as well, and they had to put more effort in converging on details. In this particular case, more effective conveyance could have helped NorthAmerica 2 to grasp the details better.

4.2. Sprint Planning

Sprint Planning was divided into two separate phases. Sprint Pre-planning meetings were held a few days before the internal Sprint Planning meetings that focused on implementation details. Internal Sprint Planning meetings were conducted independently at different sites without participation from others. The aim of the pre-planning sessions was to achieve a mutual understanding on what should be implemented in the forthcoming Sprint, similarly to Sprint Planning meetings of Scrum. Therefore, the meetings aimed for converging on details of to-be-implemented features. Pre-planning meetings were arranged so that they would fit in the ProductOwner's schedule. If the ProductOwner was not available for a scheduled meeting, the meeting would be rescheduled, if possible, so that the ProductOwner was able to participate. Table 7 indicates the communication media used between the sites during this phase.

4.2.1. Findings from NorthAmerica 1

Sprint Pre-planning meetings between NorthAmerica 1 and NorthAmerica 2 and NorthAmerica 1 and India were held separately via telephone and screen sharing software. These sessions lasted approximately one hour. The internal Sprint Pre-planning meetings at NorthAmerica 1 lasted from five to ten minutes. As it was agreed, the ProductOwner participated in Sprint Pre-planning meetings but not in internal technically oriented planning sessions.

> "I've made a conscious decision not to be too involved in internal sprint planning and try to be more involved in high-level pre-planning. Primarily because I'm not spending enough time in the market, I'm spending too much time with product development. But to be honest, where my strength lies is more in higher-level functionality, what problems are we trying to solve." (ProductOwner)

The internal planning sessions at NorthAmerica 1 were run by the MainTechLead who had extensive knowledge about technical details of the product. The following comment made by MainTechLead indicates a need for extra communication stemming from the ProductOwner's absence in internal Sprint Planning meetings:

> "A lot of times when we were thinking about the tasks that we were given, we come up with a bunch of different ways to do it in the sprint planning. And then we have to go to ProductOwner and get feedback saying we could do it in half the time if we did it this way, is that okay?"

Therefore, ***lack of involvement*** was identified as a source for this additional work. As proposed in agile approaches, this waste could

have been avoided by involving the ProductOwner in internal Sprint Planning at NorthAmerica 1.

4.2.2. Findings from NorthAmerica 2

At NorthAmerica 2, there were challenges resulting from *partially done work* that realized in a form of missing acceptance tests[3] during Sprint Pre-planning meetings: "*All the acceptance criteria should already be done before pre-planning begins. That's* (defining acceptance criteria) *not something that should be in the pre-planning session*" (NorthAmerica 2.TechLead). In the case of such events, NorthAmerica 2 conducted Spikes and re-evaluated the unclear feature. In the case that evaluating the unclear feature required a Spike, either additional information was requested or the feature was postponed to the next Sprint. If the feature could not be postponed, additional effort was added to the estimate:

> "We spike it out and say, get the information from the product owner and we'll work on it next sprint. If it's something that had to be done this sprint then sometimes we'd put a placeholder and say we're gonna do the spike and then we'll allocate five story points to do the implementation." (NorthAmerica 2.TechLead)

Earlier in the project, NorthAmerica 2 tried an approach of contacting NorthAmerica 1 via phone to sort out the problem that way. This practice was discarded since it was not seen productive. (NorthAmerica 2.TechLead):

> "We did try to phone some people to try and get information either during our lunch when NorthAmerica 1 would be available. Sometimes we started planning late so it could be overnight and we'd go to the next day, then late at night we'd try to get some information from them. When we did this at the beginning we'd actually waste the time."

From the viewpoint of communication, the amount of information provided to NorthAmerica 2 was not enough to conduct the work without obtaining additional information that could have been provided in the planning meetings by NorthAmerica 1. In the case of NorthAmerica 2, previously identified waste *partially done work* was identified as a waste. No communication-specific waste emerged.

4.2.3. Findings from India

From India, India.ProjectManager and India.Dev1 participated in these meetings. Similarly to NorthAmerica 1, insufficient time for communication was experienced during Sprint Pre-planning with India: "*I found that we have a very limited time of backlog grooming* (Sprint Pre-planning). *And in that, sometimes it happens that some of the stories were not very clear, and we end up discussing those things for a long time, and then we get less time for each to discuss*" (India.Dev1).

The following provides insights why India saw ambiguity in their user stories. The MainTechLead saw challenges in communication with India: "*It's* (communication during Spring Pre-planning) *more challenging because, it's the different team culture where it's one person* (India.ProjectManager) *doing all the talking, there's not really any conversations.*" Communication with India during the Sprint Pre-planning meetings took extra effort due to the abovementioned:

> "We have to really explain the rationale behind everything so they can understand where we're coming from. But it's harder because you're not talking to the person who's actually going to be doing the work with India. A lot of times you'd ask them a question and they'd just say, okay, sure, we'll do that. And you don't know if they actually understood it. That's why we have to be very careful with them and spend more time in carefully defining not only our questions but all the acceptance criteria." (MainTechLead)

Cultural issues are widely recognized in the GSD literature as a factor affecting communication. In this case, India lacked in-depth knowledge on the technical as well as domain aspects of the project and this was seen as a major issue: "*We really, really struggle with that*" (ProjectManager). This resulted into challenges and the biggest of them was:

> "How do we get them working on more complex features, more complex themes that we don't need to define really in detail. Like for example (a particular theme). We actually had to spend a lot of time defining that here first and then give it to them and we still have to really, really work with them." (ProjectManager)

Similar views to the lack of understanding were experienced also at India:

> "The (person at NorthAmerica 1), he doesn't have any background about the problem. So, he is not familiar or is not at the level of understanding that we have about the specific problem. So to explain the particular scenario or particular feature very clearly is a challenge." (India.QualityAssurance)

Therefore, there was a ***lack of shared understanding*** between NorthAmerica 1 and India that set increased demands for communication and made it prone to misunderstandings. This in turn required a significant communication effort in order to clarify them.

The ProductOwner (or anybody else from NorthAmerica 1) did not participate in the internal Sprint Planning meetings at NorthAmerica 2 and India. NorthAmerica 2.TechLead saw that participation from NorthAmerica 1 was not worthwhile:

> "In order for someone to attend a meeting that's four hours plus, in order to provide input for ten minutes, is not for me a good use of time. I don't see it having as much of an advantage. I wouldn't expect ProductOwner to be in there. ProductOwner's already said this is what I want done, I've agreed, you guys determine how it actually should be done. That's why we have the pre-planning."

India, however, saw benefits if someone from NorthAmerica 1 would have participated in their internal Sprint Planning meetings. India.Dev1: "*Definitely that will be beneficial for us, because sometimes it happens that we require more design or some clarification from* (NorthAmerica 1), *and then we, basically after the* (internal) *Sprint Planning we used to discuss those things in a tech call*[4] (with NorthAmerica 1)." There were requirement-related challenges in India, but due to significant time-zone difference, participation from NorthAmerica 1 was not reasonable.

Issues with technical infrastructure are recognized challenges of GSD and they were encountered also in our study. The following is not limited to this particular phase alone, but applies in all situations and phases in which communication tools were used between NorthAmerica 1 and India. Occasionally, the voice quality was poor and this resulted in challenges: "(NorthAmerica 1) *has had trouble understanding what we are talking about or we have had trouble understanding, in the sense that, because the voice quality is poor, we are not able to be sure what they have to say*" (India.ProjectManager).

Considering screen sharing software, especially over long distances, the increased lag was seen as problematic and it was sometimes significant.

[3] Acceptance criteria were the case organization's term for acceptance tests.

[4] Tech calls are discussed in more detail later in this study.

Table 8
The identified communication related waste from Sprint Planning phase.

Identified waste	Description in the context of the phase
Lack of involvement	The ProductOwner's absence in internal Sprint Planning meetings caused extra work since the most suitable way to implement something had to be agreed with the ProductOwner after the meeting.
Lack of shared understanding	Participants in NorthAmerica 1 and India did not share similar understandings of features being discussed. This resulted from the fact that India lacked deep technical and domain knowledge of the project. Also, it was difficult for India to explain their work clearly for NorthAmerica 1 since NorthAmerica 1 did not share their deep knowledge about the implementation of the features.

> "Over long distances, the lag can be frustrating." (ProductOwner)

> "We have got delays of up to 30 seconds to one minute between when I update a screen at my end and (NorthAmerica 1) is able to see the same screen. And similarly vice versa." (India.ProjectManager)

Further, the connection between the sites was unstable: "*Sometimes connection goes off, so that is also a limitation, while you reconnect, and again start a meeting. So it's sometimes time-consuming also*" (India.QualityAssurance). Despite the same communication tools being used between NorthAmerica 1 and NorthAmerica 2, technical issues were not reported. While it is unclear why communication between these sites was smooth, it can be assumed that possibly a better technical infrastructure and shorter distance between the sites could have contributed to this. The summary of identified communication related wastes is provided in Table 8.

From MST's viewpoint, the lack of shared understanding stems from insufficient conveyance since the information provided prior to pre-planning sessions was sometimes too vague. Similarly to this, missing acceptance tests mentioned by NorthAmerica 2.TechLead indicate defective conveyance. In the case of a lack of shared understanding, the issues of lacking conveyance and also convergence emerged. From a theoretical perspective, mitigation requires conveying information effectively (e.g. documenting acceptance tests properly before they are converged). In order to improve the understanding between the communicating participants, the features that are expected to be implemented should be conveyed (e.g. documented) in detail for the party with the lesser understanding.

4.3. Sprints: communication media related waste

In this section, the wastes related to communication media used during the development iterations are discussed. Table 9 depicts the use of different media between sites. The discussion of communication media between different sites is divided based on their ability to support either conveyance or convergence.

Table 9
The communication media used during Sprints phase.

Communication within NorthAmerica 1	Communication between NorthAmerica 1 and NorthAmerica 2	Communication between NorthAmerica 1 and India
Face-to-face communication, instant messaging (IM), telephone and email.	Telephone supported by screen sharing. Email.	Telephone supported by screen sharing. Email.

4.3.1. Media supporting conveyance

Email was extensively used during development. Email was seen as beneficial for more formal decisions that require a "paper trail", "*Email is really nice, if you need something a little more formal*" (PersonnelManager). Similarly to the PersonnelManager, the NorthAmerica 2.TechLead saw advantages in email: "*To have a summary of results at the end of a meeting is always handy, so to have something via email is something you can go back and refer to.*" There were, however, waste in email communication. Email is not supposed to be an effective medium for convergence, but in the project it was used also for this purpose. According to the PersonnelManager there was *handoff* in emails: "*Sometimes there's several follow-up emails that say, did you really mean this. And you compound that with the delay for each time, it's not as efficient.*"

Other stakeholders recognized *delay* in email as an issue as well: "(the challenge is) *the delay between sending it back and forth. If you would be able to get someone on the phone it's much faster to get some of your responses and explanations*" (NorthAmerica 2.TechLead). Also NorthAmerica 2.Tester mentioned *delay* as an issue in email communication: "*There's a delay in replying.*" The abovementioned is in line with findings related to delayed and complicated problem solving via email. There were also other issues in using email communication in converging information resulting from multiple and conflicting viewpoints presented in email discussions:

> "Somebody would ask him (the ProductOwner) something, but there would be other people on the emails and they'd answer actually you can't do that because technically you can't do it. So, meanwhile, the first email how would you (the ProductOwner) like us to do this. (The ProductOwner) will come back, yes, I'd like you to do that. Meanwhile there's another email that says no, actually technically you can't do it." (ProjectManager)

The ProductOwner did not answer to the latest email that had the newest information, using ***outdated information*** as the basis of making decisions. However, this waste was seen as a minor problem and the discussion converged sooner or later either by email or during a meeting. In this particular case, the reasons contributing to ***outdated information*** from the ProductOwner's side stemmed from his very busy schedule: "*His schedule is packed and I email the* (ProductOwner) *with a question, then I have no idea when I'll hear back from him.*" (UserExperience). Paradoxally, the ProductOwner's busy schedule was the reason why the majority of communication with him was conducted via email:

> "As a general rule, we'd send an email, because his (the ProductOwner) schedule is always very busy." (NorthAmerica 2.TechLead)

From India, communication with the ProductOwner was done via intermediates (mainly MainTechLead) and this was seen as a communication challenge leading to a *delay*: "*Since we don't have direct communication with the product owner, so that's basically a lag for our development. So, it basically hampers the development speed*" (India.Dev1). There was extensive email communication between the sites. Time zone difference combined with the *delay* resulting from waiting for the answer from a relevant person was seen as a challenge within India.

> "Challenges (in email communication), the turnaround time that we get for our queries. Owing to the time differences and having the people who decide what should be done, to answer our queries. So that basically leads to a delay by a day, because we have a time zone difference."(India.Dev2)

The MainTechLead saw challenges in emails sent by India. NorthAmerica 1 and India had agreed that India would prepare a daily email message explaining the current status of their work with possible questions. This information was further discussed

and clarified in a separate teleconference supported by screen sharing software: "*A lot of times their initial email to me, it may not make sense to me. So then, in my morning call, I ask them to demonstrate it, and if I'm still confused I tell them OK, make a video*" (MainTechLead). These daily calls, also referred as "Tech Calls" were held in order to compensate for the problems caused by time zone difference and limited overlapping work hours with India. These calls took place at approximately 7:30 am NorthAmerica 1 time and lasted for approximately 30 minutes.

The challenge experienced by the UserExperience in email communication provides insights to why the email communication with India was ambiguous: "*There's some time that I need to spend just making sure I have the right idea of what they're trying to communicate. There have been some miscommunications about what they were trying to get across. Just the way that the sentences are structured, it's not clear what their meaning is, so you have to kind of guess*." This language barrier led to poor conveyance.

Requirements-related communication during the Sprints with India took extra effort and was also seen as source of miscommunication (MainTechLead): "*The one thing with India is the implicit requirements that we need to be better at enumerating. This is the biggest instance of miscommunication, they're expecting everything to be very explicit*." UserExperience stated a similar communication problem with India considering the user interfaces. Everything needed to be specified in detail, since:

> "Otherwise they'll give us something weird. Even if it's just like an OK button, I still need to do a mock-up of that which takes time. Because otherwise it'll be really squished, or the buttons will be on the left side instead of the right side. It's just completely random."

The following comment by the PersonnelManager suggests that, to some extent, the need to document everything explicitly for India resulted from their lack of experience on the product: "*We have usability here, we have people that have worked with this product forever. We know the little nuances and idiosyncrasies of it and why we do things, some history behind things. They don't know that. We do things based sometimes on history*." Therefore, there was a ***lack of shared understanding*** within India that resulted in miscommunication and increased demands for conveying information.

4.3.2. Media supporting convergence

Face-to-face communication was used during the Sprints within NorthAmerica 1. While it was seen as extremely beneficial due to immediate feedback (i.e. high synchronicity and support for convergence) there were also challenges.

> "You don't have things written down. So you gotta be careful, you gotta take your notebook, write things down. MainTechLead and I will say, I'm pretty sure the ProductOwner said that, but now we gotta check, because we forgot to write it down, or neither of us can remember exactly what he said. It's hard to remember." (ProjectManager)

In this case, relying on face-to-face communication and leaving decisions undocumented led to *relearning*. Undocumented decisions themselves are a source of *information distortion*. According to the UserExperience, face-to-face communication resulted in decisions that were made too fast without thorough understanding of the topic: "*There's no time to think about things, you just have to decide. You don't get a chance to think about all the possible weird cases that could happen*." The missing details were later *converged* using email or in PlayTime Sessions described later in this paper.

Within NorthAmerica 1, there was occasional Sprint time communication with the ProductOwner via both telephone and instant messaging. Their usage was, however, limited by the ProductOwner's availability. Instant messaging was used occasionally, for example, for quickly *converging* on details and to check if the ProductOwner is at the office; "*We don't use that a ton, just once in a while when I just have a quick question or I need to know if* (ProductOwner) *is there, then I'll ask* (ProductOwner) *that*" (UserExperience). Telephone was used for converging information that would have been more laborious to converge through email:

> "I wanna discuss it (email from India) more, and it's too complicated to write. Or it would take a lot of effort to write an email, so that's when I'll see if he's there and just ask him, what do you think about this. It's just easier." (UserExperience)

4.3.3. PlayTime sessions

The PlayTime sessions were meetings during which the stakeholders of NorthAmerica 1 gathered together to use the latest build of the product together with the ProductOwner for feedback. These sessions were held from two to three times per week and were seen as an essential factor for efficient communication. India was not able to participate in these sessions due to the temporal distance. NorthAmerica 2 as well did not take part in these meetings. The reason for this was that the members of NorthAmerica 2 did not see value in participating due to the fact that the senior stakeholders steering the development and relaying information to them were located at NorthAmerica 1:

> "What I see is that (NorthAmerica 1) has the usability team, and the people who he'd (the ProductOwner) play with and try to get ideas of are there. So long as you have representation there, then that's fine. So, right now I would believe that (MainTechLead) would be our representation in this project, and anything that comes up he'll try to forward to us." (NorthAmerica 2.TechLead)

Similarly to *relearning* that emerged during face-to-face discussions outside PlayTime meetings, important details related to the feedback received from the ProductOwner were sometimes lost during PlayTime sessions:

> "It's a very informal meeting, so there's not someone taking meeting minutes. Sometimes, (the ProductOwner) is using the software and ideas are just spewing out there, and we don't capture them all, and we let them slip. We get the big things but then a lot of the smaller little details we sometimes miss and they come up in a different PlayTime. The second time we definitely write it down because we recognize it the second time." (NorthAmerica 1.MainTechLead)

This comment indicates that the information loss was not permanent due to efficient convergence mechanism that was provided by the PlayTime sessions. The importance of these sessions is illustrated by the comment from the ProductOwner: "*I think it can save a lot of time for development. You know, save them from going down a wrong path if people are there for them to ask questions and, show functionality and ask questions.*"

The developers at NorthAmerica 1 did not see any communication challenges in these meetings: "*I could only think of benefits, I couldn't think of any challenges*" (NorthAmerica 1.Dev1).

The waste identified from this phase is summarized in Table 10.

From the perspective of MST, email was used to converge information within NorthAmerica 1. The findings of this study suggest that convergence via email can cause other wastes than delay, such as ***outdated information*** in the context of this study. However, the circumstances of the project affected the use of email as a means for passing, requesting and clarifying information between the participants. Also media with high synchronicity can cause waste if the information communicated is not documented anywhere. This can, however, be mitigated by efficient convergence strategies, such as the PlayTime sessions described in this study. Also, in situations

Table 10
The identified waste from Sprints phase.

Identified waste	Description in the context of the phase
Outdated information	Email related finding. The ProductOwner did not always make his decisions based on the latest information about the particular topic.
Lack of shared understanding	India's lack of similar understanding with NorthAmerica 1 considering software's domain and design guidelines resulted in increased demands for documenting requirements assigned to them. Otherwise, the way India implemented their features was sometimes not what was expected by NorthAmerica 1.

where participants do not share a similar understanding of the product, efficient conveyance is required (as can be deducted from the tenets of MST). Efficient convergence is required to sort out misunderstandings.

4.4. Sprint Reviews

Table 11 presents the communication media used in Sprint Review sessions.

Sprint Review meetings were held for two different audiences. The Sprint Review meetings focusing on the outcome of each iteration were held face-to-face at NorthAmerica 1 when there was something to be demonstrated to the ProductOwner. Similarly to Sprint Planning and PlayTime meetings, ProductOwner's participation was mandatory and Sprint Reviews were rescheduled when necessary based on the ProductOwner's availability. However, if there was nothing to be demonstrated to the ProductOwner in NorthAmerica 1, Sprint Reviews were cancelled. The study participants from NorthAmerica 1 Sprint Reviews identified no waste.

The Sprint Reviews between NorthAmerica 1 and NorthAmerica 2 and NorthAmerica 1 and India were held via telephone and screen sharing. Similarly to NorthAmerica 1, no waste was identified. However, the challenges experienced with communication infrastructure during other phases were encountered during these meetings. From the ProductOwner's side, there were no communication issues since he was well aware of what was to be presented in each meeting:

> "Typically I know before what's been accomplished just because of my communication with the team. So I have a good understanding as to what's going to be presented." (ProductOwner)

In addition to team specific meetings, the project had joint *Master Sprint Review* meetings hosted at NorthAmerica 1 during which all the teams presented their work to other teams. However, the main purpose of these meetings was to demonstrate the software to the representatives of the case company's upper management who participated in these sessions in order to get the big picture of the project and provide their feedback. Aside from the challenges related to technical communication infrastructure, no waste was identified in these meetings.

Sprint Reviews aim to verify whether the set goals for the Sprint have been met. The demo held for the ProductOwner this session aimed towards convergence, considering the ProductOwner was well aware of the contents to be presented. Interactive media was used during all the Sprint Review sessions. This follows the suggestions of MST due to these medias strong support of synchronicity and, hence, convergence. No specific waste emerged from this phase.

Table 11
The communication media used during Sprint Reviews phase.

Communication within NorthAmerica 1	Communication between NorthAmerica 1 and NorthAmerica 2	Communication between NorthAmerica 1 and India
Face-to-face meetings.	Held via telephone supported by screen sharing software.	Held via telephone supported by screen sharing software.

Table 12
The communication media used during Retrospectives phase.

Communication within NorthAmerica 1	Communication between NorthAmerica 1 and NorthAmerica 2	Communication between NorthAmerica 1 and India
Face-to-face meetings.	NorthAmerica 1 participated via telephone supported by screen sharing software.	No involvement from NorthAmerica 1. The conclusions from India Retrospectives were sent to NorthAmerica 1 as an email.

4.5. Retrospectives

Table 12 summarizes the use of different communication media within NorthAmerica 1 and between sites during Retrospectives.

The teams held internal Retrospectives, but the ProductOwner did not participate in these. The reason for not participating was the ProductOwner's limited time:

> "Mostly it's just time. It's just a function of my time is better spent elsewhere. I'm assuming that if there's things that they need me to do, to help with the effectiveness of the sprint, they'll feed that back to me." (ProductOwner)

Instead, the MainTechLead was in charge of running the Retrospectives and participated in meetings taking place at NorthAmerica 1 and NorthAmerica 2. Due to the significant temporal distance to India, no one from NorthAmerica 1 participated in the India Retrospectives. The Retrospectives were conducted face-to-face within NorthAmerica 1.

NorthAmerica 1 participated in Retrospectives conducted within NorthAmerica 2 via telephone and screen sharing. The participation from NorthAmerica 1 was seen as beneficial:

> "It's much more productive, because we will assign them action items. A lot of the things that we need help with are not necessarily things that we can fix on our own, so we need product management in some way to take an active role in it. So we do need some sort of representation, just as long as somebody is there, I think is important." (NorthAmerica 2.TechLead)

Even though the temporal distance prevented NorthAmerica 1's participation in Retrospectives conducted in India, the potential participation of NorthAmerica 1 was not seen as beneficial, as was mentioned by India.ProjectManager:

> "Would it be helpful for someone from NorthAmerica 1 to participate, perhaps, I mean, they'd be able to see what the team thinks how the sprint went. But would it be greatly beneficial, I doubt it."

Relevant topics emerging from India Retrospectives were communicated to NorthAmerica 1 after the meetings: "*We prepare the document and send a mail* to (NorthAmerica 1). *They also look at it, and they take appropriate action and decision based on it*" (India.QualityAssurance). In summary, no waste was identified from Retrospectives phase at any site.

Within NorthAmerica 1 and between NorthAmerica 1 and NorthAmerica 2, Retrospectives were conducted using interactive media. From the MST's perspective, using media supporting convergence is in line with the propositions made in agile literature.

In the case of India, the discussions were conducted without participation from NorthAmerica 1 and action points were afterwards conveyed to NorthAmerica 1. This information was then converged later.

4.6. Waste found while discussing documentation

Documentation was used as a source of product related information. The initial themes and the user stories were stored in a dedicated Product Backlog tool. In addition, a wiki was extensively used during the development and the project had a dedicated network drive in which, for example, user interface illustrations were stored. In the wiki, there were separate "sub-wikis" for different teams. Based on the data, the following documentation-related waste was identified.

Keeping documentation up to date was seen difficult: "*The challenges are keeping them up to date at all times*" (MainTechLead). In many occasions, documentation was seen as a task of lesser priority and it had to be done later, if at all: "*The documentation usually doesn't get done later. So that's a big challenge. And it happens. We have out-of-date documentation, no question*" (PersonnelManager). This was seen as common problem that created issues for the project.

> "An outdated document or incomplete document is always the problem." (NorthAmerica 1.Dev2)

> "If the information is out of date, then it's not useful, or it might mislead you to the wrong place." (NorthAmerica 1.Dev1)

Hence, there was ***outdated information*** and it was also found that if documentation was done later, some of the important information was forgotten and not included in the documentation: "*I need to complete that* (development task) *first, then I create the document. So, sometimes it happens that you may miss some information or you may miss some knowledge to share in that document*" (India.QualityAssurance). This is time related information distortion.

In addition, ***restricted access to information*** was found for the project wiki at India:

> "We don't have overall access for all the pages. This limits us for the pages which we are working at. If we need designs from other team's page, we inform this to (MainTechLead), and (MainTechLead) sends our read-only access to that page[5]." (India.Dev1)

Relevant information was scattered across different documents stored at the project wiki: "*I can look on the wiki and I can show all sorts of implementation, design documents, detail design documents, functional specs, that sort of thing. But nowhere in one document are all these things concentrated in one spot*" (PersonnelManager).

This was seen as resulting from the fact that there was no uniform approach to update the project wiki: "*There's no process that says you must do it* (updating information) *this way. If people don't sort of look at it and follow that same form, you can get stuff in there that gets lost*" (PersonnelManager).

This comment indicates that finding relevant information was also difficult. The comment from NorthAmerica 2.Dev supports this: "*The search feature is terrible. It's almost unusable to search it, I find, unless you actually can find someone who can tell you where it is, it's hard to find documentation.*" Information was sometimes also fragmented. It was mentioned that information related to features they were implementing was documented in the wikis related to the earlier versions of the product, which in turn can make finding the relevant information difficult: "*Some information is documented in* (the earlier version) *wiki. For new people that join* (the project), *he doesn't know where to search for the information*" (NorthAmerica 1.Dev1). Further, the organization of the wiki caused problems in form of limited visibility to interrelated features implemented by different sites:

> "There's the documentation for features which exist and the usability wiki pages, which are linked to ours, but we don't necessarily see the changes that they're doing on their wiki pages, so we don't get that visibility. Something that may affect us may be documented in another team's wiki page, but you don't know to look there." (NorthAmerica 2.TechLead)

Therefore, the abovementioned issues can be considered as ***scattered information***.

Table 13 summarizes the communication waste found while discussing documentation.

From the perspective of MST, documentation is an efficient medium for *conveying* information. In this case, the identified waste indicates also defective *conveyance*. The ability of documents to convey information would improve if they would be kept updated, made readily available for every participant, and stored cohesively.

Table 13
The identified waste found while discussing documentation.

Identified waste	Description in the context of the phase
Outdated information	The documentation was outdated and was missing relevant information. This resulted in defective conveyance of information.
Restricted access to information	India did not have access to all information stored in project wiki. The access to relevant information had to be obtained via NorthAmerica 1. This waste is different from ***limited access to information***, which is related to the existence of information. In the case of this waste, the relevant information exists, but is not available.
Scattered information	Relevant information was scattered across several documents and it was difficult to find. This consumed resources.

4.7. Proposed corrective actions to the identified waste

Table 14 discusses the solution proposals for the identified waste in the context of this study. We provide generalized descriptions of the wastes and the corrective actions are derived from both empirical findings and existing recommendations found from literature. The actions stemming from literature are indicated by appropriate references.

4.8. Threats to validity

According to Yin (2003), validity of case studies can be approached from the perspective of construct validity, internal validity, external validity and reliability. In the following, we discuss the validity of our study based on this classification.

Construct validity: in order to improve construct validity, multiple data sources were utilized during the study. In addition, every action related to the study was documented in a form of field notes and a research diary, thus establishing a chain of evidence (Yin, 1994). Finally, ambiguities in data were validated, when possible, with the informants as suggested in Yin (1994). We also applied *investigator triangulation* proposed in Stake (1995) and discussed results and interpretations with colleagues, in this case between the researchers of this study, to prevent the problem of *multiple realities* (Kaplan and Duchon, 1988; Goetz and LeCompte, 1984).

[5] This comment is an edited version of the original. Grammatical errors are removed and the comment is edited for readability.

Table 14
Proposed corrective actions to the identified waste in the context of the case project.

Waste	Generalized description derived from the findings	Proposed corrective actions for the case company in the context of the project
Lack of involvement	The absence of key stakeholders from the process phases where their participation is essential to acquire information and provide input for the development and/or receive feedback.	Active collaboration and communication between all project participants is emphasized (Beck, 2000). Therefore, all the teams should be involved in the development process from the beginning. In addition, the stakeholders should participate in activities requiring their presence.
Lack of shared understanding	The communicating participants do not share similar understanding and expertise on the topic being communicated. This creates increased demands for communication and makes it prone to misunderstandings.	Considering the tenets of MST (Dennis et al., 2008), unclear topics should be communicated in as much detail as possible using media supporting conveyance. In this particular case, detailed specifications should be written for India and the information should be converged actively. Effective convergence strategy was applied with India by daily Tech Calls. This approach is in line with the recommendation of using domain experts communicating daily with distributed teams (Summers, 2008).
Outdated information	The topic that is being communicated or required is not based on the latest information about it.	In the context of email, this waste was a minor problem in this study and the waste was mitigated either in subsequent emails or in a meeting, such as the PlayTime session. In this case, the mechanisms for mitigating this waste were adequate. Considering documentation, ensuring that it remains up-to-date should be paid attention. This could be achieved by adding documentation related aspects as a part of the acceptance criteria in order to complete a task.
Restricted access to information	Relevant information is not readily available for all parties that need it.	Provide appropriate access rights to all participants from the beginning of the project.
Scattered information	The information related to the product or the project is dispersed in several locations which make it difficult and time-consuming to find.	Establish uniform policies to store and document information and follow these guidelines.

Internal validity: issues related to internal validity arise mainly when causal relations are examined. According to Robson (2002), causal relationships are often used as a tool in explanatory studies when seeking an explanation of a situation or a problem. Case studies are primarily used in exploratory studies, which aim to understand what is happening and seeking new insights (Robson, 2002). However, due to limited time reserved for the study, the effects of proposed solutions were not verified. Also, it could not be observed if the solutions would have generated additional waste themselves.

External validity: the results are drawn from a particular context in which the Product Owner is collocated with senior project members that work as intermediates with the distributed teams. Results can be valid only in this or similar contexts. However, the context itself is an important factor in case studies (e.g. Benbasat et al., 1987). In addition, interpretive case studies do not seek generalizability (Orlikowski and Baroudi, 1991).

Reliability: in order to improve reliability of the study, every action and item (e.g. codes and interview questions) related to the study was documented and, if necessary, updated as the study proceeded to form a chain of evidence.

There were additional limitations. The waste mitigation approach presented in this paper does not take into account against what criteria the improvements should be measured. Metrics such as defect rates, the amount of change requests, and perhaps the most essential agile metric, Velocity (Schwaber, 2004), could provide some guidelines for assessing the impacts of the waste mitigation actions. As mentioned earlier, waste identification can be seen as a challenge in software development since waste is not so visible, nor as well understood as in the manufacturing industry. In addition, the use of Media Synchronicity Theory was limited to conveyance and convergence processes alone, and the use of different media was observed through them. However, the purpose of this study was not to study the theory itself, but use its main concepts as the lens for analysing communication.

Theoretical saturation of data is a measure that is reached after new themes, insights or issues are not emerging from the data that is being gathered (Strauss and Corbin, 1998). The lack of practical guidelines that assist the estimation of sample sizes (e.g. the amount of people to be interviewed) has been criticized, e.g. in Guest et al. (2006). In order to answer to this critique, Guest et al. conducted a study in which they concluded that saturation was reached during the first twelve interviews they conducted within a homogenous sample, i.e. the participants were chosen according to common criteria. In our study, the participants were considered homogenous for all being software professionals that worked on the same globally distributed agile software development project. A total of 14 people were interviewed during the study, so considering the results presented in Guest et al. (2006), the data seems to be saturated. Whether saturation was achieved during data collection is unknown, due to limited availability of the participants. However, convenience sampling is commonplace in software engineering studies due to limited stakeholder availability for interviewing (Lethbridge et al., 2005).

5. Discussion and future work

In addition to the identified communication wastes, the issues previously encountered in globally distributed environments, also encountered in this study, can be considered wasteful. Challenges with *language barrier* have been discussed in Layman et al. (2006), Uy and Ioannou (2008) and Kajko-Mattsson et al. (2010), and problems with technical communication infrastructure in Ågerfalk (2004), Williams and Stout (2008) and Therrien (2008). Communication delays resulting from temporal distance are widely reported as challenges in distributed environments (e.g. Holmstrom et al.,

2006; Ågerfalk and Fitzgerald, 2006; Conchúir et al., 2009; Sarker and Sahay, 2004; Ågerfalk, 2004; Boland and Fitzgerald, 2004). We deliberately excluded these from being communication specific waste, since these challenges are well-known issues in global software development. Also, some communication media are inherently wasteful. For example, email has a built-in delay resulting from its asynchronous nature, where resolving even simple matters can take a significant time converging when communicated via email (Carmel and Agarwal, 2001). Similar findings were present in our study as well. Despite these downsides, email is a dominant communication tool in globally distributed settings and it was extensively used in this project as well. The Product Owner had a very busy schedule and this resulted in using email as the main communication media between him and the rest of the development organization. Hence, certain conditions may dictate the use of a medium that is not necessarily the best possible option given the communication requirements.

This study indicated that some of the wastes presented in Poppendieck and Poppendieck (2007) and Mandić et al. (2010) were also found in this study. Considering the impacts of these wastes and the wastes related to communication, their realization did not create any problems that would have seriously jeopardized the project. The only major challenge was the communication with India, but this caused only increased demands for conveying and converging information. Since the case project had established an efficient communication strategy with India in a form of daily calls, this mitigated the effects of waste identified from communication with them. Considering the findings of this study, they are in line with the findings made by Ikonen et al. (2010) who concluded that software projects can be successful despite existing waste.

For example, ***scattered information*** required more work to obtain the necessary, and available, information so that misunderstandings stemming from ***outdated information*** in email discussions were corrected at some point during the development. ***Outdated information*** in the context of documentation was, however, more permanent in nature. Even though seen as a challenge, it did not create any enduring problems throughout the development. However, if documents are used to convey information and convergence on this information is defective, making decisions based on ***outdated information*** can possibly have serious effects. ***Restricted access to information*** did not create major problems since India was able to obtain the information after requesting access to it. However, in the study reported in Korkala et al. (2010) the required information remained restricted from the organization needing it due to bureaucratic reasons. Therefore, even though the problems in this study were minor, ***restricted access to information*** can have more severe impacts on the project. ***Lack of involvement*** was encountered in two phases of the project. Within NorthAmerica 1 it caused communication overhead due to ProductOwner's absence in internal Sprint Planning sessions and with India it resulted in *extra effort*. India was not able to participate in PlayTime sessions due to temporal distance. From this viewpoint, ***lack of involvement*** was also a *type 1* waste. Existing literature shows that ***lack of involvement*** can have other effects as well. Korkala et al. (2006) studied four agile development projects with varying customer involvement during the development. The results show that, while customer's involvement during development decreases the amount of defects, these defects could have been avoided by increased, regular customer feedback.

The most significant pattern related to waste was the communication challenges between NorthAmerica 1 and India. Considering the interaction between NorthAmerica 1 and India, ***lack of shared understanding*** emerged as the most significant waste encountered during Sprint Pre-planning sessions and during the Sprints. The increased need for communication was identified by NorthAmerica 1 representatives before the study was even initiated. In addition, a ***lack of involvement*** in the case of India could have contributed to ***lack of shared understanding*** since India was not properly aware of end-user needs. A previous finding from Korkala et al. (2010) supports this: the lack of domain knowledge combined with unelaborated requirements provided by an uninvolved customer organization caused problems. In addition to a significant gap in knowledge of the domain and the knowledge related to the product itself, barriers in language and culture contributed to impacts of this waste by increasing the difficulty of communication. Therefore, our finding provides additional evidence on the importance of understanding cultural distance (Noll et al., 2010). Further, the identified waste can fall into both waste categories. As an example, ***restricted access to information*** can be a *type 1* waste that cannot be removed due to company policies. This further emphasizes the context-dependent nature of the mitigation strategies. Practices from agile development methods, suggestions for MST, and more rigid policies for ensuring that documentation remains updated and information is readily available for all, can provide guidelines for mitigating the effects of waste. While the policies are presented as empirical solutions, they follow the suggestions of Boehm and Turner (2003) about balancing agility with more discipline when seen necessary.

In general, the project was able to maintain active communication and followed the agile principle of face-to-face communication when it was applicable. PlayTime sessions with mandatory Product Owner participation can be seen as a factor contributing to the agile principle of active customer involvement and fast feedback and guidance from the customer side. Considering communication in the project as a whole, it was active and involved all the participating sites using interactive media when possible. This can be seen as a communication policy following the agile recommendation of interactive and active communication. In addition, the effective communication strategy within the project helped to mitigate the effects of identified wastes.

There are several opportunities for future research. It would be useful to evaluate if the communication wastes identified in this study are valid and if other wastes can be identified. This could be studied by applying the approach presented in this work in other contexts as well. We believe that this would further help companies to identify communication waste in their development efforts and to provide them tools to mitigate them. The proposed waste identification process requires additional research as well. In this particular case, we applied the process in the context of a single development project. Applying this approach in the context of a larger development programme consisting of several development projects would be a potent arena to test and further develop the waste identification process. In addition, identifying wastes and the ways of mitigating them could potentially have significant impacts for both industry and research. It would be interesting to apply the presented approach to identify wastes from areas outside communication, such as requirements management and defect correction, or from other levels within an organization.

Suitability of the mitigating process should be studied further since we were not able to verify the impacts of proposed improvement suggestions. It would also be very interesting to study communication wastes in globally distributed environments through other processes and approaches as well, since this could provide additional evidence on the validity of the findings presented in this work. Socio-technical congruence (Cataldo et al., 2006, 2008) could be used as such a process. According to Perry et al. (1994) the amount of communication developers engage in during the development is significant and communication on its behalf is a central mechanism for coordination (Kraut and Streeter, 1995). Socio-technical congruence focuses on the "fit" between task dependencies and the individuals' coordination activities. From the perspective presented by Kraut and Streeter (1995) this translates

to how and what people communicate. According to Cataldo et al. (2006), congruence reduces the time required to perform different tasks while the use of different communication channels are chosen to better fit the task at hand. In addition, when there is a proper fit between the ways of coordination and the needs for coordination, the time for resolving modification needs is significantly reduced (Cataldo et al., 2008). Hence, socio-technical congruence framework supports more effective communication, as does our approach. Therefore, applying socio-technical congruence could provide more insights to communication in globally distributed agile development and could perhaps validate whether the waste identified in this study are valid in software engineering.

6. Conclusions

Communication has been widely recognized as one of the key elements contributing to the success or failure of a software development project. Nowadays, implementing software in a globally distributed fashion is a common approach and this creates additional challenges for communication. For example, temporal, cultural and geographical distances (Noll et al., 2010) and their combination (Holmstrom et al., 2006) introduce challenges to the success of distributed efforts. More traditional plan-driven development approaches rely on formal communication (e.g. detailed extensive documentation) for conveying information, while agile development approaches emphasize informal communication relying on face-to-face interactions instead. Effective communication is difficult to achieve in distributed plan-driven efforts alone and agile approaches create additional challenges since significant geographical and temporal distances can prevent face-to-face, or other interactive communication.

Existing literature has approached this challenge by attempting to create suggestions establishing and maintaining effective communication in globally distributed agile development projects (e.g. Layman et al., 2006; Kircher et al., 2001; Danait, 2005). While they are valuable contributions to the topic, they do not provide companies the means to analyze and improve communication in their globally distributed agile efforts. To address this shortcoming, we conducted a case study within a North American software intensive company that was implementing a product across three sites in a globally distributed fashion. We constructed a waste identification process through which communication between the key stakeholders was analyzed, using the concept of waste from lean manufacturing (Ohno, 1988) and Lean software development (Poppendieck and Poppendieck, 2007). In addition to finding waste already identified in literature (Poppendieck and Poppendieck, 2007; Mandić et al., 2010), we identified five wastes that were specific to communication and presented the case company solution proposals for tackling them in the context of their project. While the improvement actions are dependent on the context, these wastes can provide companies an idea of what could be the non-value adding elements in communication within their globally distributed agile projects.

We defined three research questions through which we approached this work. The answer to the sub-research question aiming to identify the waste in communication within globally distributed agile development projects is the five wastes of communication. These wastes are ***lack of involvement***, ***lack of shared understanding***, ***outdated information***, ***restricted access to information*** and ***scattered information***. The second sub-question aimed to identify means for identifying waste. The answer to this question is provided in a form of the waste identification approach described in this work. Companies and researchers can use this approach by applying the same actions conducted in this study. Next, we describe how the waste identification approach could be conducted based on this study. The answer to the main research question will be provided after this description.

First, the project from which the communication waste is to be identified and mitigated is selected. The next step is to identify the key stakeholders that would participate in the waste identification process. After this, the development approach taken in the chosen project and the key steps in which communication takes place is identified. In this study the chosen development approach was Scrum and the all the key steps of this method (Sprint Planning, Sprint, Sprint Review and Retrospective) were present. In addition, we analyzed communication before the implementation phase begun, with communication related to documentation included in the analysis. Also appropriate inputs through which communication is analyzed and waste is extracted should be identified. In this study, we used a set of questions to attain a cohesive view of communication, including both positive and wasteful aspects. In addition, the documentation provided by the case company supported us to understand the development approach. Hence, the companies or other instances should apply all the data seen relevant for understanding communication within the chosen project. Reaching saturation in the data collection (i.e. analysis of the data collection did not reveal new information) was used as the measure for obtaining all information necessary for understanding communication and identifying waste. The main output from the steps is the communication wastes. Finally, the measures for removing or mitigating the wastes should be identified and these measures incorporated into the process. In our case, these measures were either empirical or derived from literature. This process should be conducted at regular intervals.

The main research question of this study is to identify how waste identification can improve communication in globally distributed agile software development. The answer to this question is that waste identification, when completed in a structured manner, can point to non-value producing communication elements and the waste can then be mitigated. The results of this study provide companies an idea of the potential wastes that may be present in their globally distributed agile efforts, as well as a technique for mitigating their effects.

For the research community, this study contributes to the field of communication in globally distributed agile software development by presenting and discussing five wastes specific to communication. Also, the presented waste identification approach is the initial step towards a systematically validated approach for analysing and improving communication in a globally distributed agile development context.

References

Ågerfalk, P., Fitzgerald, B., 2006. Flexible and distributed software processes: old petunias in new bowls? Commun. ACM 49 (10), 10–27.
Ågerfalk, P.J., 2004. Investigating actability dimensions: a language/action perspective on criteria for information systems evaluation. Interact. Comput. 16 (5), 957–988.
Battin, R., Crocker, R., Kreidler, J., Subramanian, K., 2001. Leveraging resources in global software development. IEEE Softw. 18 (2), 70–77.
Beck, K., 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley, Upper Saddle River, NJ, USA.
Benbasat, I., Goldstein, D.K., Mead, M., 1987. The case research strategy in studies of information systems. MIS Quart. 11 (3), 369–386.
Berger, H., 2007. Agile development in a bureaucratic arena—a case study experience. Int. J. Inform. Manage. 27 (6), 386–396.
Boehm, B.W., Turner, R., 2003. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Professional, Boston, MA, USA.
Boland, D., Fitzgerald, B., 2004. Transitioning from a co-located to a globally-distributed software development team: a case study at Analog Devices Inc. In: 3rd International Workshop on Global Software Development (ICSE2004), 23–28 May 2004, Scotland, UK, pp. 4–7.
Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3 (2), 77–101.
Carmel, E., Agarwal, R., 2001. Tactical approaches for alleviating distance in global software development. IEEE Softw. 18 (2), 22–29.

Cataldo, M., Herbsleb, J.D., Carley, K.M., 2008. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In: Proceedings of ESEM'08, 9–10 October 2008, Kaiserslautern, Germany, pp. 2–11.

Cataldo, M., Wagstrom, P.A., Herbsleb, J.D., Carley, K.M., 2006. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In: Proceedings of CSCW'06, 4–8 November 2006, Banff, Alberta, Canada, pp. 353–362.

Ceschi, M., Sillitti, A., Succi, G., De Panfilis, S., 2005. Project management in plan-based and agile companies. IEEE Softw. 22 (3), 21–27.

Conchúir, E.Ó., Ågerfalk, P.J., Olsson, H.H., Fitzgerald, B., 2009. Global software development: where are the benefits? Commun. ACM 52 (8), 127–131.

Corbin, J., Strauss, A., 2008. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Thousand Oaks, CA, USA.

Cruzes, D.S., Dybå, T., 2011. Research synthesis in software engineering: a tertiary study. Inform. Softw. Technol. 538 (5), 440–455.

Daft, R.L., Lengel, R., Trevino, L.K., 1987. Message equivocality, media selection, and manager performance: implications for information support systems. MIS Quart. 11 (3), 355–366.

Daft, R.L., Lengel, R.J., 1986. Organizational information requirements, media richness and structural design. Manage. Sci. 32 (5), 554–571.

Damian, D., Moitra, D., 2006. Guest Editors' introduction: Global software development: how far have we come? IEEE Softw. 23 (5), 17–19.

Damian, D., Zowghi, D., 2003. Requirements engineering challenges in multi-site software development organizations. Requir. Eng. J. 8, 149–160.

Danait, A., 2005. Agile offshore techniques—a case study. In: Proceedings of Agile Development Conference (AGILE 2005), 24–29 July 2005, Denver, CO, USA, pp. 214–217.

DeLuca, D., Valacich, J.S., 2006. Virtual teams in and out of synchronicity. Inform. Technol. People 19 (4), 323–344.

Dennis, A.R., Fuller, R.M., Valacich, J.S., 2008. Media, tasks, and communication processes: a theory of media synchronicity. MIS Quart. 32 (3), 575–600.

Dennis, A.R., Valacich, J.S., Speier, C., Morris, M.G., 1998. Beyond media richness: an empirical test of media synchronicity theory. In: Proceedings of HICSS'98, 6–9 January 1998, Kohala Coast, Hawaii, USA, pp. 48–57.

Drummond, B.S., Francis, J., 2008. Yahoo! Distributed agile: notes from the world over. In: Proceedings of Agile 2008, 4–8 August 2008, Toronto, ON, Canada, pp. 315–321.

Ebert, C., De Neve, P., 2001. Surviving global software development. IEEE Softw. 18 (2), 62–69.

Eisenhardt, K.M., 1989. Building theories from case study research. Acad. Manage. Rev. 14 (4), 532–550.

Glaser, B., Strauss, A., 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Publishing Company, New Jersey, NY, USA.

Goetz, J.P., LeCompte, M.D., 1984. Ethnography and Qualitative Design in Educational Research. Academic Press, Orlando, FL, USA.

Gorton, I., Motwani, S., 1996. Issues in co-operative software engineering using globally distributed teams. Inform. Softw. Technol. 38 (10), 647–655.

Guest, G., Bunce, A., Johnson, L., 2006. How many interviews are enough? An experiment with data saturation and variability. Field Methods 18 (1), 59–82.

Herbsleb, J.D., Mockus, A., Finholt, T.A., Grinter, R.E., 2001. An empirical study of global software development: distance and speed. In: Proceedings of ICSE2001, 12–19 May 2001, Toronto, ON, Canada, pp. 81–90.

Herbsleb, J., Grinter, R., 1999. Splitting the organization and integrating the code: Conway's Law revisited. In: Proceedings of ICSE'99, 16–22 May 1999, Los Angeles, CA, USA, pp. 85–95.

Herbsleb, J., Moitra, D., 2001. Global software development. IEEE Softw. 18 (2), 16–20.

Hicks, B.J., 2007. Lean information management: understanding and eliminating waste. Int. J. Inform. Manage. 27 (4), 233–249.

Holmström, H., Conchuir, E.O., Agerfalk, P.J., Fitzgerald, B., 2006. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In: Proceedings of ICGSE'06, 16–19 October 2006, Costão do Santinho, Florianópolis, Brazil, pp. 3–11.

Holmstrom, H., Fitzgerald, B., Ågerfalk, P.J., Conchuir, E.O., 2006. Agile practices reduce distance in global software development. Inform. Syst. Dev. 23 (3), 7–18.

Ikonen, M., 2010. Leadership in Kanban software development projects: a quasi-controlled experiment. In: Proceedings of LESS2010, 17–20 October 2010, Helsinki, Finland, pp. 85–98.

Ikonen, M., Kettunen, P., Oza, N., Abrahamsson, P., 2010. Exploring the sources of waste in Kanban software development projects. In: Proceedings of EUROMICRO2010, 1–3 September 2010, Lille, France, pp. 376–381.

Kajko-Mattsson, M., Azizyan, G., Magarian, M.K., 2010. Classes of distributed agile development problems. In: Proceedings of Agile 2010, 9–13 August 2010, Orlando, FL, USA, pp. 51–58.

Kaplan, B., Duchon, D., 1988. Combining qualitative and quantitative methods in information systems research: a case study. MIS Quart. 12 (4), 571–586.

Kircher, M., Jain, P., Corsaro, A., Levine, D., 2001. Distributed eXtreme programming. In: Proceedings of XP2001, 21–23 May 2001, Villasimius, Sardinia, Italy, pp. 66–72.

Komi-Sirviö, S., Tihinen, M., 2005. Lessons learned by participants of distributed software development. Knowl. Process Manage. 12 (2), 108–122.

Korkala, M., Pikkarainen, M., Conboy, K., 2010. Combining agile and traditional: customer communication in distributed environment. In: Ågerfalk, P.J, Smite, D. (Eds.), Agility Across Time and Space. Springer, Berlin, Heidelberg, pp. 201–216.

Korkala, M., Abrahamsson, P., 2007. Communication in distributed agile development: a case study. In: Proceedings of EUROMICRO2007, 28–31 August 2007, Lübeck, Germany, pp. 203–210.

Korkala, M., Abrahamsson, P., Kyllönen, P., 2006. A case study on the impact of customer communication on defects in agile software development. In: Proceedings of Agile 2006, 23–28 July 2006, Minneapolis, MN, USA, pp. 76–88.

Kraut, R.E., Streeter, L.A., 1995. Coordination in software development. Commun. ACM 38 (3), 69–81.

Layman, L., Williams, L., Damian, D., Bures, H., 2006. Essential communication practices for Extreme Programming in a global software development team. Inform. Softw. Technol. 48 (9), 781–794.

Lee, S., Yong, H.S., 2010. Distributed agile: project management in a global environment. Empir. Softw. Eng. 15 (2), 204–217.

Lethbridge, T.C., Sim, S.E., Singer, J., 2005. Studying software engineers: data collection techniques for software field studies. Empir. Softw. Eng. 10 (3), 311–341.

Mandić, V., Oivo, M., Rodríguez, P., Kuvaja, P., Kaikkonen, H., Turhan, B., 2010. What is flowing in lean software development? In: Proceedings of LESS2010, 17–20 October 2010, Helsinki, Finland, pp. 72–84.

Melnik, G., Maurer, F., 2004. Direct verbal communication as a catalyst of agile knowledge sharing. In: Proceedings of Agile 2004, 22–26 June 2004, Salt Lake City, UT, USA, pp. 21–31.

Miles, M.B., Huberman, A.M., 1994. Qualitative Data Analysis: An Expanded Sourcebook, 2nd ed. SAGE Publications Inc., Thousand Oaks, CA, USA.

Mockus, A., Herbsleb, J., 2001. Challenges of global software development. In: Proceedings of METRICS 2001, 4–6 April, London, England, pp. 182–184.

Nerur, S., Mahapatra, R.K., Mangalaraj, G., 2005. Challenges of migrating to agile methodologies. Commun. ACM 48 (5), 72–78.

Niinimäki, T., Piri, A., Lassenius, C., Paasivaara, M., 2010. Reflecting the choice and usage of communication tools in GSD projects with media synchronicity theory. In: Proceedings of ICGSE 2010, 23–26 August 2010, Princeton, NJ, USA, pp. 3–12.

Noll, J., Beecham, S., Richardson, I., 2010. Global software development and collaboration: barriers and solutions. ACM Inroads 1 (3), 66–78.

Nöteberg, A., Benford, T.L., Hunton, J.E., 2003. Matching electronic communication media and audit tasks. Int. J. Account. Inform. Syst. 4 (1), 27–55.

Ohno, T., 1988. Toyota Production System: Beyond Large-scale Production. Productivity Press, Cambridge, MA, USA.

Orlikowski, W.J., Baroudi, J.J., 1991. Studying information technology in organizations: research approaches and assumptions. Inform. Syst. Res. 2 (1), 1–28.

Paasivaara, M., Durasiewicz, S., Lassenius, C., 2008. Distributed agile development: using scrum in a large project. In: Proceedings of ICGSE 2008, 17–20 August 2008, Bangalore, India, pp. 87–95.

Perry, D.E., Staudenmayer, N.A., Votta, L.G., 1994. People, organizations, and process improvement. IEEE Softw. 11 (4), 36–45.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J., 2008. The impact of agile practices on communication in software development. Empir. Softw. Eng. 13 (3), 303–337.

Poppendieck, M., Poppendieck, T., 2007. Implementing Lean Software Development: From Concept to Cash. Addison-Wesley Professional, Boston, MA, USA.

Poppendieck, M., Poppendieck, T., 2003. Lean Software Development: An Agile Toolkit, 1st ed. Addison-Wesley, Upper Saddle River, NJ, USA.

Robson, C., 2002. Real World Research, 2nd ed. Blackwell, Oxford, UK.

Royce, W.W., 1970. Managing the development of large software systems. In: Proceedings of IEEE Wescon, August 1970, Los Angeles, CA, USA.

Sarker, S., Sahay, S., 2004. Implications of space and time for distributed work: an interpretive study of US–Norwegian systems development teams. Eur. J. Inform. Syst. 13 (1), 3–20.

Schwaber, K., 2004. Agile Project Management with Scrum. Microsoft Press, USA.

Schwaber, K., Beedle, M., 2002. Agile Software Development with Scrum. Prentice-Hall, Upper Saddle River, NJ, USA.

Seaman, C.B., 2002. Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. 25 (4), 557–572.

Stake, R., 1995. The Art of Case Research. Sage Publications, Thousand Oaks, CA, USA.

Strauss, A., Corbin, J., 1998. Basics of Qualitative Research: Grounded Theory Procedures and Techniques, 2nd ed. Sage Publications, Thousand Oaks, CA, USA.

Summers, M., 2008. Insights into an agile adventure with offshore partners. In: Proceedings of Agile 2008, 4–8 August 2008, Toronto, ON, Canada, pp. 333–338.

Sureshchandra, K., Shrinivasavadhani, J., 2008. Adopting agile in distributed development. In: Proceedings of ICGSE 2008, 17–20 August 2008, Bangalore, India, pp. 217–221.

Sutherland, J., Viktorov, A., Blount, J., Puntikov, N., 2007. Distributed scrum: agile project management with outsourced development teams. In: Proceedings of HICSS2007, 3–6 January 2007, Waikoloa, Big Island, HI, USA, p. 274.

Therrien, E., 2008. Overcoming the challenges of building a distributed agile organization. In: Proceedings of Agile 2008, 4–8 August 2008, Toronto, ON, Canada, pp. 358–372.

Uy, E., Ioannou, N., 2008. Growing and sustaining an offshore Scrum engagement. In: Proceedings of Agile 2008, 4–8 August 2008, Toronto, ON, Canada, pp. 345–350.

Vax, M., Michaud, S., 2008. Distributed agile: growing a practice together. In: Proceedings of Agile 2008, 4–8 August 2008, Toronto, ON, Canada, pp. 310–314.

Wallach, E.J., 1983. Individuals and organizations: the cultural match. Train. Dev. J. 37 (2), 29–36.

Williams, W., Stout, M., 2008. Colossal, scattered, and chaotic (planning with a large distributed team). In: Proceedings of Agile 2008, 4–8 August 2008, Toronto, ON, Canada, pp. 356–361.

Womack, J.P., Jones, D.T., 1996. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Simon & Schuster, New York, NY, USA.

Yin, R.K., 2003. Case Study Research, Design and Methods, 3rd ed. Sage Publications, Beverly Hills, CA, USA.
Yin, R.K., 1994. Case Study Research Design and Methods. Sage Publications, Thousand Oaks, CA, USA.

Mikko Korkala works as a research scientist at VTT Technical Research Centre of Finland. He has worked with agile methodologies since 2001 and is currently finalizing his Ph.D on customer communication in distributed agile software development. Prior to joining VTT at 2007 he has worked in the University of Oulu, Finland as an assisting teacher and as a researcher. He has also worked as a software engineer in the industry. In addition to scientific research, he has provided agile trainings for over 400 software professionals and held talks about agile software development and lean software development in Finland and other countries. Further, he has worked as an onsite agile coach and Scrum Master and has outlined agile processes for companies. In addition to agile software development, his interests include lean software development, service development, and innovation in the context of software industry.

Frank Maurer is a professor of computer science and associate vice-president (research) at the University of Calgary. His research interests include application engineering for digital surfaces, analytics & visualization and agile software methodologies. Dr Maurer leads the NSERC SurfNet Strategic Research Network. SurfNet is a Canadian research alliance of academic researchers, industry partners, and government collaborators. The goal of SurfNet is to improve the development, performance, and usability of software applications for surface computing environments: non-traditional digital display surfaces including multi-touch screens, tabletops, and wall-sized displays.

Series title and number

VTT Science 80

Title	**Customer communication in distributed agile software development**
Author(s)	Mikko Korkala
Abstract	Agile software development methods emerged in the late 1990s and early 2000s with a promise to deliver high quality software within schedule and budget. One of the key differences between so-called traditional development approaches and agile methods is that agile software development methods put significant emphasis on communication. In agile development, communication is proposed to be conducted in an informal face-to-face manner. This communication extends beyond the development team, involving all project stakeholders including the customers, whose role in agile development is pivotal. Since their emergence, agile methods have been adopted in distributed development environments at sites that can be separated by significant geographic and temporal as well as cultural distances. Communication in distributed environments is already difficult in more traditional development projects, and it is even more challenging in agile development projects that emphasise face-to-face communication. The focus of this thesis is to understand how customer communication can be improved in distributed agile software development. The research problem is approached from the perspectives of customer involvement, actions for communication improvement and distributed context through a series of case studies. The empirical evidence is derived from five different case studies involving both small-scale efforts and large, globally distributed development projects. The findings of this thesis have both theoretical and practical implications. The first implication for research comprises the five wastes of communication; lack of involvement, lack of shared understanding, outdated information, restricted access to information and scattered information. These wastes provide a unique view to communication hindrances that are present in distributed agile software development. The second theoretical implication is that lack of trust between the distributed partners is potentially the single most important obstacle to customer communication. As a practical implication, this study provides a toolbox that can be used in order to improve customer communication in distributed agile software development. In this work, the toolbox is first defined on the basis of existing literature and then further complemented with the findings of the studies. The toolbox presents different communication challenges and the solution proposals for them. Based on this study, two distinct areas emerged from the toolbox. These themes are the customer's involvement in the process and systematic analysis and improvement of customer communication, both of which should be given additional attention in distributed agile efforts.
ISBN, ISSN	ISBN 978-951-38-8230-3 (Soft back ed.) ISBN 978-951-38-8231-0 (URL: http://www.vtt.fi/publications/index.jsp) ISSN-L 2242-119X ISSN 2242-119X (Print) ISSN 2242-1203 (Online)
Date	April 2015
Language	English, Finnish abstract
Pages	123 p. + app. 77 p.
Name of the project	
Commissioned by	
Keywords	Agile software development, distributed agile software development, customer communication, communication challenges, communication waste, waste, communication solutions
Publisher	VTT Technical Research Centre of Finland Ltd P.O. Box 1000, FI-02044 VTT, Finland, Tel. 020 722 111

Julkaisun sarja ja numero

VTT Science 80

Nimeke	**Asiakaskommunikaatio hajautetussa ketterässä ohjelmistokehityksessä**
Tekijä(t)	Mikko Korkala
Tiivistelmä	1990-luvun lopussa ja 2000-luvun alussa ilmaantuneet ketterät ohjelmistokehitysmenetelmät painottavat aktiivista kommunikaatiota kaikkien kehitysprojektiin osallistuvien tahojen välillä. Kommunikaation näkökulmasta keskeisempänä erona niin kutsuttuihin perinteisiin kehitysmenetelmiin verrattuna on se, että ketterissä menetelmissä kommunikaatio on jatkuvaa ja epäformaalia, kun taas perinteisissä menetelmissä eri osapuolten välinen kommunikaatio on epäsäännöllistä ja noudattaa formaaleja menetelmiä. Ketteriä ohjelmistokehitysmenetelmiä sovelletaan nykyään myös hajautetussa ohjelmistokehityksessä, mikä asettaa kehitystyölle omat haasteensa huomattavienkin maantieteellisten etäisyyksien, aikaerojen sekä kulttuurillisten eroavaisuuksien muodossa. Nämä puolestaan saattavat aiheuttaa huomattavia kommunikaatiohaasteita. Tämän tutkimuksen tarkoituksena on tarkastella, kuinka asiakaskommunikaatiota voidaan parantaa hajautetussa ketterässä ohjelmistokehityksessä. Tätä tutkimusongelmaa lähestytään asiakkaan osallistuttamisen, asiakaskommunikaation parantamisen ja hajautetun ohjelmistokehityksen näkökulmista. Tutkimus on suoritettu viitenä erillisenä tapaustutkimuksena, jotka lähestyvät asiakaskommunikaatiota sekä pienten kehitysprojektien että laajojen globaalien hankkeiden näkökulmista. Tämän tutkimuksen keskeinen käytännön tulos on kokoelma työkaluja, jotka auttavat tunnistamaan hajautettujen ketterien ohjelmistokehitysprojektien kohtaamia kommunikaatiohaasteita. Tämä kokoelma tarjoaa myös ratkaisumalleja näiden haasteiden lieventämiseen. Työssä esiteltävä työkalukokoelma on johdettu olemassa olevasta kirjallisuudesta ja sitä on täydennetty tämän tutkimuksen tuloksilla.

Tutkimuksen keskeinen teoreettinen tulos on viisi kommunikaatiohukkaa: aktiivisen osallistumisen puute (lack of involvement), yhteisymmärryksen puute (lack of shared understanding), vanhentunut tieto (outdated information), rajoitettu pääsy tietoon (restricted access to information) sekä hajallaan oleva tieto (scattered information). Nämä kommunikaatiohukat tarjoavat uuden näkökulman hajautetuissa ketterissä kehitysprojekteissa kohdattaviin kommunikaatiota haittaaviin tekijöihin. Lisäksi tämä tutkimus esittää, että projektiin osallistuvien osapuolten välinen luottamuksen puute on mahdollisesti tärkein yksittäinen tekijä, joka vaikeuttaa asiakaskommunikaatiota hajautetuissa ketterissä ohjelmistokehityshankkeissa. |
ISBN, ISSN	ISBN 978-951-38-8230-3 (nid.) ISBN 978-951-38-8231-0 (URL: http://www.vtt.fi/publications/index.jsp) ISSN-L 2242-119X ISSN 2242-119X (Painettu) ISSN 2242-1203 (Verkkojulkaisu)
Julkaisuaika	Huhtikuu 2015
Kieli	Englanti, suomenkielinen tiivistelmä
Sivumäärä	123 s. + liitt. 77 s.
Projektin nimi	
Rahoittajat	
Avainsanat	ketterä ohjelmistokehitys, hajautettu ketterä ohjelmistokehitys, asiakaskommunikaatio, kommunikaatiohaasteet, kommunikaatiohukka, hukka, kommunikaatioratkaisut
Julkaisija	Teknologian tutkimuskeskus VTT Oy PL 1000, 02044 VTT, puh. 020 722 111

Customer communication in distributed agile software development

One of the key differences between so-called traditional development approaches and agile methods is that agile software development methods put significant emphasis on communication. In agile development, communication is proposed to be conducted in an informal face-to-face manner. This communication extends beyond the development team, involving all project stakeholders including the customers, whose role in agile development is pivotal. Since their emergence, agile methods have been adopted in distributed development environments at sites that can be separated by significant geographic and temporal as well as cultural distances.The focus of this thesis is to understand how customer communication can be improved in distributed agile software development.

The findings of this thesis have both theoretical and practical implications. The first implication for research comprises the five wastes of communication; lack of involvement, lack of shared understanding, outdated information, restricted access to information and scattered information. These wastes provide a unique view to communication hindrances that are present in distributed agile software development. The second theoretical implication is that lack of trust between the distributed partners is potentially the single most important obstacle to customer communication. As a practical implication, this study provides a toolbox that can be used in order to improve customer communication in distributed agile software development. In this work, the toolbox is first defined on the basis of existing literature and then further complemented with the findings of the individual case studies. The toolbox presents different communication challenges and the solution proposals for them.

ISBN 978-951-38-8230-3 (Soft back ed.)
ISBN 978-951-38-8231-0 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN-L 2242-119X
ISSN 2242-119X (Print)
ISSN 2242-1203 (Online)