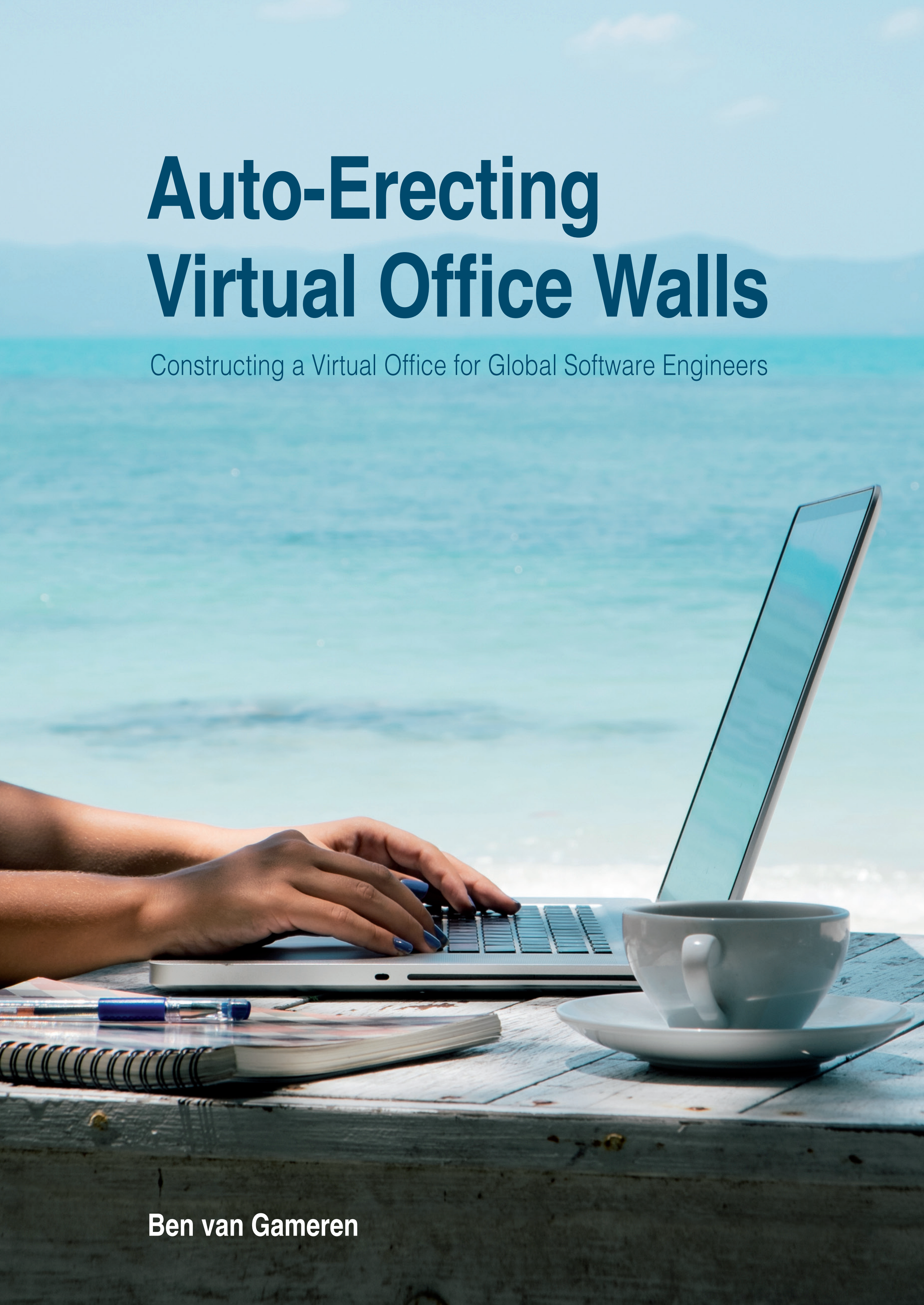
# Auto-Erecting Virtual Office Walls

Constructing a Virtual Office for Global Software Engineers



Ben van Gameren

# Auto-Erecting Virtual Office Walls

Constructing a Virtual Office for Global Software Engineers

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 17 juni 2014 om 12:30 uur
door

Benjamin Jacobus Abraham van GAMEREN

Master of Science in Computer Science
geboren te Rotterdam.

ISBN: 978-94-6186-317-1

Author email: `benvangameren@gmail.com`

*"In the long history of humankind those who learned to collaborate and improvise most effectively have prevailed"*

Charles Darwin

# Acknowledgements

The last four years, in which I pursued my PhD, are an amazing source of inspiration, joy and knowledge. It all started back in the summer of 2009 during a relaxing holiday in Mallorca. During this holiday I had to make the important decision whether or not to accept the opportunity to continue my academic career at the Delft University of Technology. At that time I just successfully completed my Master of Science project, in which I took my first steps in research. During this project I closely collaborated with Kevin Dullemond and under the supervision of Rini van Solingen I published my first paper. So, on a lovely beach in Mallorca I decided to accept this exciting opportunity, although I never expected to pursue an academic career. Now, four years later, I wish to thank all those who have supported me on this journey.

First of all, I would like to express my gratitude to Rini for giving me the opportunity to pursue my PhD under his excellent supervision. During the many discussions we had, he always gave his honest and professional opinion to provide guidance on how to conduct research properly. I am very thankful for his enthusiastic and human approach, which make him a great person to work with. Also, I would like to thank my other promotor, Arie van Deursen, for always offering a critical yet constructive opinion. His open and interested attitude make him easily approachable.

Furthermore, I would like to thank all my colleagues at IHomer who have participated and supported me during the course of this research. They provided me with valuable feedback, enthusiastic ideas, and were always interested in the research itself. Especially, I would like to thank Dick Stegeman for his positive and encouraging attitude towards our research. I really enjoyed conducting research at IHomer and I am proud to be part of this company.

Subsequently, I would like to thank Kevin Dullemond. During our PhD studies Kevin and I closely collaborated and spent a lot of time together. We had many interesting discussions in which we formulated theoretical concepts and refined new ideas. I really enjoyed this complementary collaboration and had a great time. Furthermore, I also enjoyed the experience of both visiting multiple international conferences together and exploring places I have never been before.

Next, I would like to thank the members of my defense committee: David Redmiles, Darja Smite, Egon Berghout, Henk Sips and Patricia Lago for reviewing my dissertation and providing valuable feedback.

Finally, I would like to thank my family and friends. I would like to thank Kim Stehouwer for applying his creative skills to the design of the cover of this dissertation. I would like to thank my parents and mother-in-law for their unconditional support and confidence in me. Whatever happens I can always rely on you. I also would like to thank my sister, Linda, for her support and all the fun and crazy times. Last but certainly not least, I would like to thank Sabrina. Dear Sabrina, thank you for always being there for me. Your love allows me to give the best of myself and I really enjoy the time we spend together. During the last years we have gained a lot of good memories, including our campervan holiday to New Zealand, and I cannot wait to see what the future holds for us. I love you!

*Delft,* *Ben van Gameren*
*June 2014*

# Contents

# Part I

# Prologue

Chapter 1

# Introduction

Software engineering is a highly collaborative activity in which knowledge about the context in which one is working is essential to properly collaborate with others [Sch02, Syr97]. With information about the context we mean information about the other team members, their activities, information about the state of the project and so on. This kind of information is necessary for coordinating actions, managing transactions between individual and shared work, discussing tasks, anticipating others' actions, and finding help [Sch02, Syr97, Gut02]. In scientific literature the term *'awareness'* is often used to denote this [Sch02, Dou92]. Dourish and Bellotti more formally define awareness as: *"An understanding of the activities of others which provides a context for your own activity"* [Dou92].

In the traditional co-located office setting this information is exchanged relatively passively and unobtrusively [Sch02, Fog05]. This is because all information is available in a single place, the office building, and is accessible by all employees present at that location. In such an environment team members are frequently able to both see and hear each other, as such it is relatively easy for software engineers to acquire and sustain a shared understanding.

However, nowadays, software engineering is increasingly conducted outside of the traditional co-located office building. Software is for instance developed in multiple dislocated office buildings or even from home. This is the result of the increasing globalization of business [Car99, Her01, Her07] and the rising popularity of working from home [Die09]. Global software engineering introduces multiple benefits and challenges. Benefits of the globalization of business include: market-proximity [Gri99, Dam06], reducing time-to-market by working around the clock [Car99, Ebe01], flexibility with respect to business opportunities [Car99, Her99], reducing costs by delegating work to countries with low labor cost [Car01, Dam06] and being able to fully utilize available resources [Her01, Dam06]. Working from home also introduces benefits, including: increased autonomy [Har02], increased flexibility [Har02], increased productivity [Hes91], increased motivation [Pra93] and improvement in the quality of the environment [Har02].

In contrast with these benefits, global software engineering also introduces a number of challenges in relation to communication, coordination and control of the software development process [Car01]. For example, lack of informal communication [Car99, Her99, Her01, Åge05], reduced hours of collaboration [Bat01, Kie03, Hol06, Åge08], communication delay [Åge05, Her05, Her07, Con06], and loss of cohesion [Car99, Her03, Her07].

In a distributed environment team members are not sharing a physical work environment and are outside of sensory range of each other. Therefore it becomes infeasible to exchange information without some kind of technological support. In the last decades the (global) software engineering community has developed many technological solutions to support globally dispersed teams in performing their tasks. These solutions are in general inferior to the way contextual information is shared in a traditional co-located setting, in the sense that in comparison it (i) takes more effort because the communication is more intentional [Gut04], (ii) is more obtrusive [Fog05], (iii) happens less frequently [Her03, All77] and (iv) contains less information [Gut04, Her99]. As such we can conclude that spreading awareness is more difficult in a distributed setting.

The research presented in this dissertation continues upon the insight that many aspects of awareness that are disseminated as a natural by-product of co-located working are difficult to achieve in distributed working environments [Omo10]. Therefore it is important that technological support is developed to fully exploit the advantages of global software engineering. As such, the main goal of our research is:

**Research Goal**

> *"To support global software engineers with technological support for aiding them to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities"*

In this chapter we first provide a brief overview of our vision on how best to support global software engineers to acquire a sufficient level of awareness for their work activities, in section 1.1. Next, in section 1.2 we motivate why such research is important and how it could impact today's society. Subsequently, in section 1.3, we present the research goal of this dissertation and outline our main research questions. In section 1.4, we briefly describe the empirical research strategy we used to answer the research questions. Next, in section 1.5 we present the two organizations which played an important role in our research. In section 1.6, we describe the origin of the chapters of this dissertation. Finally, in section 1.7, we discuss the research division and responsibilities, since part of the research reported in this dissertation is the result of a joint effort.

## 1.1 A Virtual Office

In the traditional co-located setting information to achieve awareness is available in a single place, the office building, and is accessible by all employees present at that location. In such a co-located setting information is exchanged relatively passively and unobtrusively [Sch02, Fog05]. But, how are software engineers capable of abstracting useful information without experiencing an overload of information? Probably this has mostly to do with the design of the office building [All07]. In general an office building consists of multiple rooms, for example a foyer, a kitchen, meeting rooms and offices. All of these rooms have their own characteristics and by moving around in the building and selecting a room which characteristics match the software engineer's needs, an engineer is able to change the context of his activities. By organizing their work environment in such a way, software engineers can easily spread awareness between all involved stakeholders.

In a distributed setting, developers no longer share a physical work environment and as a consequence cannot exchange information without technological support. So, in order to collaborate with their colleagues engineers need to use technological solutions to be able to retrieve information relevant to their current task. The software engineering community has developed several solutions to fulfill this need. However most of these solutions only support a specific type of information and this information cannot be processed by other solutions directly [PR12]. Therefore, engineers need to manually analyze the available information to be able to acquire the information they need.

To be able to acquire awareness in a relatively passive and unobtrusive fashion, such as in the co-located setting, we need to automate this analytical process of accessing, combining and filtering the available information. In essence we need to automate the process of restricting the available information to that information an actor needs to carry out his current activity. We propose to call this mechanism a *'Virtual Office Wall'* [Gam12]. Filtering information in such a fashion resembles the creation of *'moving'* office walls in a co-located setting which move around based on the work an engineer is carrying out. When such a mechanism which introduces logical boundaries is successfully applied to a virtual office it even has the potential to outperform the co-located office setting.

## 1.2 Motivation

Research regarding how best to support global software engineering teams is becoming part of the body of knowledge of global software engineering. This is because decreasing the impact of distance on software engineering work is a long lasting quest within the global software engineering community. Currently, research focuses on reducing the importance of having knowledge about the context of the project, for example by modularizing the work and thus reducing the amount of distributed collaboration. Such an approach however only attempts to cope with the problems and not to solve them. The research presented in this dissertation will attempt to do so by targeting the lack of awareness in a distributed setting itself.

From a societal point of view this research also has the potential to provide a breakthrough, because it could solve an important issue in today's society: daily polluting traffic jams. If this research manages to alleviate the challenges faced when working distributed, people no longer have to commute to their work on daily basis. This could lead to less traffic congestion, a reduced carbon dioxide footprint, and, in a business context, the opportunity to work on several tasks at once, while still being experienced by colleagues as a full team member.

## 1.3 Problem Statement

From the previous sections we can conclude that a mechanism is needed which automatically regulates the available information to that information a software engineer needs to carry out his current activity. Such a mechanism introduces logical instead of fixed boundaries and successfully applying it in a global software engineering team enables acquiring awareness in a relatively passive and unobtrusive fashion. As such, the main goal of this dissertation is: *"To support global software engineers with technological support for aiding them to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities"*.

In an attempt to reach this goal we study three important aspects of the design, implementation and evaluation of such technological support: (i) constructing a virtual office, (ii) communicating in a virtual office, and (iii) information needs in a virtual office.

### 1.3.1 Constructing a Virtual Office

In this part we formulate both our vision on how to provide distributed software engineers with a sufficient level of awareness and an approach to construct such technological solutions.

#### Virtual Office Walls

Software engineers need information about the context in which they are working to be able to carry out their work activities. In a distributed setting, software engineers use several technological solutions to fulfill this need. However, the majority of these solutions only support a single aspect of the development process. As a result, each engineer has to manually analyze, filter and combine the available information in order to acquire the context of their current work activity. Manually analyzing, filtering and combining available information can however be quite time-consuming. Therefore, we focus on how best to support distributed software engineers with the context of their current work activity.

**Research Question 1**

*What are the requirements for technological support to provide distributed software engineers with the context of their current work activity?*

To answer this first research question, we first provide our vision on how to provide distributed software engineers with a sufficient level of awareness for their work activities. In chapter 2, we discuss our vision on how auto-erected virtual office walls can help distributed software engineers to relatively passively and unobtrusively acquire a context of their work activities. We elaborate on this concept of virtual office walls and discuss the prerequisites which should be fulfilled to construct such mechanisms. Next, we discuss how these prerequisites can be fulfilled. Finally, we perform a focus group [Kon04] to evaluate the applicability of the proposed theoretical concept in an industrial setting.

Next to formulating our vision, it is also important to formulate a feasible approach to develop, implement and validate solutions which fulfill our vision. In chapter 3, we present a research approach to (i) identify real-life global software engineering problems, (ii) propose and implement solutions for these problems, and (iii) evaluate these solutions in an industrial setting. Such an approach provides us with the empirical data we need to validate the impact of the proposed solution, and helps us in pinpointing important open research challenges.

Both our vision on how best to support global software engineers, and our approach to develop, implement and validate such solutions are important aspects in answering the first research question. We will use the concept of virtual office walls in answering the other research questions and to elaborate on our findings. This enables us to derive a set of requirements a virtual office should fulfill. These requirements also contribute to answering the first research question, because they provide important guidelines on how best to support global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities. These requirements are presented in the last section of each chapter of this dissertation, and an overview of all identified requirements is given in chapter 10.

### 1.3.2 Communicating in a Virtual Office

In this part we look at the value of communication in global software engineering. We both research the value of overhearing conversations in a distributed setting, and the value of mood sharing in such a setting.

#### Overhearing Conversations

Conversations between software engineers are an important source of information. Conversations help software engineers to integrate and coordinate their work, share knowledge about the actual work, and create new knowledge. Besides having conversations, also overhearing conversations of others is useful. In a distributed setting technological solutions to have conversations are commonly used, however overhearing conversations of others is not explicitly supported. Therefore, we identify the following research question:

**Research Question 2**

*What is the value of overhearing conversations in global software engineering?*

To answer the second research question, we first provide a theoretical motivation why the overhearing of conversations of others is valuable to a distributed software engineering team, in chapter 4. We provide a definition of a conversation which is applicable in the context of global software engineering. Subsequently we both discuss the various uses conversations have in collaborative work and the benefits of being able to overhear conversations of others. Finally we provide a definition of an *'Open Conversation Space'*, a space in which both having conversations and overhearing conversations of others is possible, and present a set of requirements such a space should fulfill.

Next, in chapter 5, we empirically evaluate the value of the concept of overhearing conversations in the field of software engineering. In this empirical study we investigate whether researching how to enable overhearing conversations in a distributed setting is worth pursuing. To acquire the empirical data we perform both a focus group [Kon04] and a questionnaire [Fin03]. In the focus group we identify: (i) the benefits and challenges of having insight in active conversations, (ii) the important types of information about a conversation, (iii) the actions possible on a conversation, and (iv) the benefits and challenges of having access to the finished conversations. Following this we perform a questionnaire to determine the relative importance of these benefits, challenges, information items and possible actions. In this study we critically analyze the benefits and challenges of overhearing conversations of others. As such we are able to determine whether research about support for overhearing conversations is worth pursuing.

Finally, in chapter 6, we present a technological implementation which enables the overhearing of conversations in a distributed setting and explain how it fulfills the requirements of an open conversation space. We design and implement this technological solution to be able to perform an empirical case study to measure the value of overhearing conversations in global software engineering from actual industrial experience. In this study we use four methods to acquire the empirical data we need: a focus group [Kon04], a semi-structured interview [Fon05], a questionnaire [Fin03], and transactional log analysis [Jan08].

The theoretical motivation why the overhearing of conversations is valuable, the evaluation of the value of overhearing conversation in the field of software engineering, and the evaluation of the value of overhearing conversation in the field of global software engineering all contribute to answering the second research question.

#### Mood Sharing

Distributed software engineers face the challenge of staying connected because they no longer see each other on a daily basis. Therefore we believe both sharing a small amount of information and sharing mood information is essential to alleviate challenges of this nature. On the one hand sharing a small amount of information is essential because being able to exchange such information makes people feel more connected. On the other hand mood sharing is essential because being aware of the emotional state of colleagues makes it possible to act accordingly. Therefore, we formulate the following research question:

**Research Question 3**
*What is the value of microblogging with mood-indicators in global software engineering?*

In an attempt to answer the third research question we conduct a study to understand how microblogging with mood-indicators helps distributed organizations in knowledge sharing. This study is presented in chapter 7. In this study we use a microblogging solution extended with mood indicators to research (i) the topics discussed in such a solution, (ii) the impact of the introduction of a such a solution on a software team, (iii) the impact of the distribution on the use of such a solution, and (iv) how team composition impact collaboration with such a solution. We collect the empirical data we need for this study by mining over a year of usage data of such a microblogging solution. First we code the content of all posts and comments of this solution, and based on the results of this analysis we perform semi-structured interviews [Fon05] with distinctive users of the system. The findings of this empirical study directly contribute to answering the third research question.

### 1.3.3 Information Needs in a Virtual Office

In the last part we look at the information needs of global software engineers. We research what information global software engineers want to know immediately, during which activities they prefer to be or prefer not to be interrupted, and the impact of automating the process of restricting the available information to that information a software engineer needs to carry out his current activity.

#### Information Needs

Distributed software engineers have to manually analyze, filter and combine available information in order to acquire a sufficient level of awareness without experiencing an overload of information. Therefore it seems beneficial to construct a mechanism which automatically determines what information is relevant when performing a collaborative activity. This, however, does not imply software engineers should be immediately informed of this information because they could be performing an activity during which they prefer not to be interrupted. Therefore, we identify the following research question:

**Research Question 4**

> *How to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer?*

To answer the fourth research question, we conduct an Estimate-Talk-Estimate study [Gus73] with experienced software engineers. This study is presented in chapter 8, and focuses on how to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer. Therefore, we are interested in a list of information items software engineers immediately want to be informed about, and in a list of activities during which software engineers prefer not to be interrupted. The outcomes of this study could be contradicting because it is likely both of these lists contain at least one item. Therefore, we are also interested in what information software engineers want to know immediately, even though they are performing an activity during which they prefer not to be interrupted. The findings of this study provide valuable insights on how to regulate information available to software engineers and answer the fourth research question.

### Virtual Office Walls

A virtual office wall is a mechanism which automatically regulates information to support distributed software engineers in performing collaborative activities. These walls reduce the available information to only that information which is relevant to the current activity of an engineer. As such, these walls have the potential to increase the actual and perceived speed and accuracy of the activities carried out. We formulate the following research question to study the value of the presence of virtual office walls:

**Research Question 5**

> *What is the value of automating the process of restricting the available information to that information a software engineer needs to carry out his current activity?*

We conduct a controlled experiment [Woh00] with experienced software engineers as study participants to answer the last research question, see chapter 9. In this experiment we try to find out how valuable virtual office walls are for real-life distributed software engineers during their day-to-day activities. We research whether there is a relation between the presence of virtual office walls and the actual and perceived speed and accuracy of the work carried out by the participants. Additionally, we measure the extent in which the participants experience the presence of virtual office walls as useful.

## 1.4 Research Approach

To answer our research questions, we have conducted multiple studies. In most of these studies the identification of real-life problems, the development of the proposed solutions and/or the evaluation of the proposed solutions were conducted in close cooperation with global software engineering companies. Colin Potts proposed the *industry as laboratory* approach to refer to a setting in which the industrial setting is used as a test environment [Pot93]. Collaborating with industrial organizations has benefits. Firstly, employees of these companies have quite a good understanding of what is needed to improve the current situation. Secondly, we can also perform high quality evaluations because these companies match the target setting for which we are attempting to solve issues: distributed organizations. Finally, employees of these companies benefit directly from the proposed solutions, since they encounter the issues we are attempting to solve in their daily work.

To fully exploit the opportunities of this collaboration, we have used the following research strategy in each of the studies we conducted:

1. Identify a real-life global software engineering problem
2. Propose and implement a solution for this problem
3. Evaluate the solution with experienced people from industry

The first step in our research strategy is to identify and select a real-life global software engineering problem. This selection is made in collaboration with our industrial partners and is based on the theoretical and practical value of solving the problem at hand. Next, we conduct a thorough study to learn about the problem and propose a suitable solution for this problem. In this exploratory phase of our research approach we used (empirical) data gathering techniques to acquire the information needed. On the one hand we studied the existing literature to map out the current research area. Therefore, we specified a number of search strings using relevant terms based on the current research questions, and looked for relevant publications which met these criteria. Next to these publications itself we also looked for relevant references in these publications to ground the current research area in literature. On the other hand we used empirical data gathering techniques; focus groups [Kon04], semi-structured interviews [Fon05], questionnaires [Fin03], and an Estimate-Talk-Estimate study [Gus73] to acquire relevant information from industry. Subsequently, we implement a solution for the problem in close collaboration with our industrial partner and conduct an empirical evaluation to study the applicability of the proposed solution in industry. We used multiple data gathering techniques to acquire the data needed to investigate the applicability of the proposed solution. Next to the data gathering methods used in the exploratory phase of our research; we mined and coded user data, conducted transactional log analysis [Jan08] to analyze user behavior, and performed a controlled experiment [Woh00]. After completing this research strategy, this approach is repeated and a new global software engineering problem is selected.

## 1.5 Industrial Setting

We have conducted different studies to answer the research questions of this dissertation. In most of these studies the identification of real-life problems, the development of the proposed solution and/or the evaluation of the proposed solution was conducted in close cooperation with two industrial partners: IHomer and Exact Software. IHomer is a Dutch software engineering company founded in August of 2008. The company currently employs 21 people and is distributed in the true sense of the word. This is because the default work location of the employees is their home. As a consequence, the people are quite experienced with dealing with the difficulties of working distributed from each other. This makes the company a particularly suitable setting for performing our research. During his PhD study the author has been part of this company. As such, he experienced real-life global software engineering problems, proposed and implemented solutions for these problems, and evaluated these solutions in this or another industrial setting. His work for IHomer consisted of researching the global software engineering problems, allowing him to fully devote himself to identify and solve the problem at hand. So, to be clear his work did not consist of doing billable work for the customers of IHomer. The main advantage of this collaboration is that the author possesses insight knowledge of a setting which experiences the problems of global software engineering and is able to assist such settings in handling these problems.

Our other industrial partner is Exact Software. Exact Software is a software development company operating in 40 countries. It offers Enterprise Resource Planning software for medium-sized and small businesses. At the end of 2012 it employed approximately 1700 employees worldwide. The specific group of employees involved in our study on the value of overhearing conversations in global software engineering worked on a product called Exact Online. The majority of the people in this group, approximately 70 people, worked out of the office location in Delft (The Netherlands) and was co-located on a single floor. However, also three people from the Wemmel (Belgium) office participated as well as two from the Minneapolis (USA) office. Next to this, people worked from home fairly often and frequently communicated using Instant Messaging software even when working from the same office.

Finally, we also conducted an Estimate-Talk-Estimate study. In such a study participants are asked to provide reasons for their decisions and to respond to the decisions made by the other participants. It is essential that the members of this study have different backgrounds so they can provide each other with new information and revise their opinions based on this information. Therefore we selected participants from nine distributed organizations, ranging from small to large size enterprises.

## 1.6 Origin of Chapters

The main chapters of this dissertation, chapters two to nine, are strongly based on previously published papers at workshops and conferences on software engineering. Since these peer-reviewed publications were published independently, all chapters are self-contained and have their own individual contributions. There is however some redundancy in the background material, motivation, and examples. In addition all chapters end with a *'Concluding Remarks'* section. In this section we briefly summarize the research presented in this chapter, present our view on how this chapter contributes to the research goal of this dissertation, and provide requirements of a virtual office.

This dissertation consists of three parts: (i) constructing a virtual office, (ii) communicating in a virtual office, and (iii) information needs in a virtual office. In the first part, constructing a virtual office, we provide an approach which can be used to research how best to support awareness in global software engineering. In the second part we look at the value of communication in global software engineering. In this part we research both the value of overhearing conversations, and the value of microblogging with mood indicators in global software engineering. Finally in the third part of this dissertation we research the information needs of global software engineers. In this part we research both what information software engineers want to know immediately, and when they prefer not to be interrupted. We conclude this part by studying the value of automating the process of restricting the available information to that information a software engineer needs to carry out his current activity.

#### Part I: Constructing a Virtual Office

The chapters in this part are strongly based on the following two publications:

- Chapter 2 is based on our publication *"Auto-Erecting Virtual Office Walls"* in the Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing [Gam12].
- Chapter 3 is based on our publication *"Supporting Distributed Software Engineering in a Fully Distributed Organization"* in the Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering [Dul12b].

#### Part II: Communicating in a Virtual Office

This part consists of three chapters which are based on the research presented in the following four publications:

- Chapter 4 is based on both our publication *"Virtual Open Conversation Spaces: Towards Improved Awareness in a GSE Setting"* in the Proceedings of the 5th International Conference on Global Software Engineering [Dul10] and on our publication *"Overhearing Conversations in Global Software Engineering - Requirements and an Implementation"* in the Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing [Dul11c].
- Chapter 5 is based on our publication *"An Exploratory Study on Open Conversation Spaces in Global Software Engineering"* in the Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing [Dul11b].
- Chapter 6 is based on both our publication *"Overhearing Conversations in Global Software Engineering - Requirements and an Implementation"* in the Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing [Dul11c] and on our publication *"An Industrial Evaluation of Technological Support for Overhearing Conversations in Global Software Engineering"* in the Proceedings of the 7th International Conference on Global Software Engineering [Dul12a].
- Chapter 7 is based on our publication *"Fixing the 'out of sight out of mind' problem - One Year of Mood-Based Microblogging in a Distributed Software Team"* in the Proceedings of the 10th International Workshop on Mining Software Repositories [Dul13b].

#### Part III: Information Needs in a Virtual Office

The chapters in this part are slight adaptations of the following two publications:

- Chapter 8 is based on our publication *"When to Interrupt Global Software Engineers to Provide them with What Information"* in the Proceedings of the 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing [Gam13b].
- Chapter 9 is based on our publication *"Auto-Erecting Virtual Office Walls a Controlled Experiment"* in the Proceedings of the 8th International Conference on Global Software Engineering [Gam13a].

## 1.7 Research Division and Responsibilities

During his PhD study the author of this dissertation closely collaborated with Kevin Dullemond. Both researchers studied related research subjects in the field of global software engineering, which resulted in multiple joint research publications. The main advantage of this joint approach is that the research subjects could be investigated more in-depth and complete.

The author of this dissertation fulfills the role of main contributor for all publications included in this dissertation. However, part of the research reported in this dissertation is a joint effort of Kevin Dullemond and the author of this dissertation. In this joint work both authors are fully responsible for defining the scope of the research, implementing the approach used, conducting the empirical evaluation, and writing the scientific publications. To make the division of the research and responsibilities explicitly clear, we specify for each of the publications included in this dissertation whether the author of this dissertation is the main contributor, or whether Kevin Dullemond and the author of this dissertation are equal contributors. In the latter case the two authors are listed in alphabetical order, both in the published scientific papers, and in the following division of research and responsibilities.

The author of this dissertation is the main contributor of the following publications:

- The publication *"Auto-Erecting Virtual Office Walls"* which is co-authored by Kevin Dullemond and Rini van Solingen [Gam12].
- The publication *"When to Interrupt Global Software Engineers to Provide them with What Information"* which is co-authored by Rini van Solingen [Gam13b].
- The publication *"Auto-Erecting Virtual Office Walls a Controlled Experiment"* which is co-authored by Rini van Solingen and Kevin Dullemond [Gam13a].

Kevin Dullemond and the author of this dissertation are equal contributors of the following publications:

- The publication *"Supporting Distributed Software Engineering in a Fully Distributed Organization"* which is co-authored by Kevin Dullemond and Rini van Solingen [Dul12b].
- The publication *"Virtual Open Conversation Spaces: Towards Improved Awareness in a GSE Setting"* which is co-authored by Kevin Dullemond and Rini van Solingen [Dul10].
- The publication *"An Exploratory Study on Open Conversation Spaces in Global Software Engineering"* which is co-authored by Kevin Dullemond and Rini van Solingen [Dul11b].
- The publication *"Overhearing Conversations in Global Software Engineering - Requirements and an Implementatoin"* which is co-authored by Kevin Dullemond and Rini van Solingen [Dul11c].
- The publication *"An Industrial Evaluation of Technological Support for Overhearing Conversations in Global Software Engineering"* which is co-authored by Kevin Dullemond [Dul12a].

- The publication *"Fixing the 'out of sight out of mind' problem - One Year of Mood-Based Microblogging in a Distributed Software Team"* which is co-authored by Kevin Dullemond, Margaret-Anne Storey and Arie van Deursen [Dul13b].

Part II

# Constructing a Virtual Office

Chapter 2

# Auto-Erecting Virtual Office Walls

*Collaborative software engineering is increasingly carried out from multiple, physically separated, locations around the globe. Software engineers are no longer tied to a fixed workplace and have the opportunity to work from the location of the customer, their home and even from their holiday location. When working in such a distributed setting, software engineers also need information about the context in which they are working to be able to collaborate effectively with their colleagues. In the last decades multiple technological solutions were developed by the software engineering community to fulfill this need. However, the majority of these solutions only support a single aspect of the development process, so each software engineer has to manually analyze, filter and combine the available information in order to acquire a sufficient level of awareness. Manually analyzing, filtering and combining available information can however be quite time-consuming and therefore we focus on how to automate this process. In this chapter we present our vision on how auto-erected virtual office walls can help distributed software engineers to relatively passively and unobtrusively accomplish this automation.*

This chapter is strongly based on our publication *"Auto-Erecting Virtual Office Walls"* in the Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2012). This publication is co-authored by Kevin Dullemond and Rini van Solingen

## 2.1 Introduction

In collaborative work, awareness is essential to properly collaborate with your colleagues [Sch02, Syr97]. With awareness we mean the information which is necessary to provide software engineers with the context in which they are working. Examples of such information items are: information about the other members in the project team, their activities, and information about the current state of the project. Dourish and Bellotti more formally define awareness as [Dou92]: *"An understanding of the activities of others which provides a context for your own activity"*. For software engineers it is essential to have a sufficient level of awareness, because software engineering is a collaborative activity which requires engineers to coordinate their efforts to be able to produce a functional system.

However, both due to the globalization of business [Car99, Her01, Her07] and due to the fact that people work from home more and more [Die09], people no longer share a physical work environment and as a consequence cannot exchange information without technological support. So, in order to collaborate with colleagues in a distributed setting, technological support is required to be able to acquire and maintain awareness. In the last decades the (global) software engineering community has developed many technological solutions to support globally dispersed teams in performing their tasks. Portillo-Rodríguez et al. [PR12] provide a systematic mapping review of available tools in the field of global software engineering and what functionality these tools offer. Several of the tools discussed are widely adopted by distributed development teams and provide the team members with information. Most of these solutions only support a single aspect of the development process and as a consequence many diverse tools are needed to provide the user with all the information he or she needs. Accordingly, all this information needs to be analyzed, combined and filtered manually by each software engineer to acquire the information necessary to create the context of his current activity. However, this process can be quite time-consuming, therefore we focus on how to automate this process. As such the main goal of this chapter is: *"To find out how auto-erected virtual office walls can help distributed software engineers to acquire the context of their current activity"*

In section 2.2 we define virtual office walls and introduce two prerequisites of the construction of these. Next, in section 2.3, we look at the first prerequisite and discuss that both access to data from a wide variety of tools is needed and the means to integrate it to create valuable information. Subsequently, in section 2.4, we look at the second prerequisite and discuss a way to describe the context of a software engineer. In section 2.5, we validate our representation of the context of a software engineer in an industrial setting. Finally, we present conclusions and discuss opportunities for future work in section 2.6.

## 2.2 Virtual Office Walls

As discussed in the introduction, software engineering is a collaborative activity which requires potentially many engineers to coordinate their actions to be able to produce a system. In order to coordinate their actions engineers need to spread awareness among each other. In the traditional co-located setting all information is available in a single place, the office building, and is accessible by all employees present at that location. In such a co-located setting awareness is spread relatively passively and unobtrusively [Sch02, Fog05]. But, how are the engineers capable of abstracting useful information without experiencing an overload of information? Probably this has mostly to do with the design of the office building [All07]. In general an office building consists of several rooms, for example a foyer, a kitchen, meeting rooms and offices. All of these rooms have their own characteristics; the meeting room, for example, has several attributes which facilitate group discussions such as a white board, a beamer and the room's size. By moving around in the building and selecting a room which characteristics match the software engineer's needs, an engineer is able to change the context of his activities. Another example is that people who work on related tasks are often seated in close proximity to each other. By organizing their work environment in such a way, they can easily spread awareness between all involved stakeholders. So, when working in a co-located setting, software engineers are continuously aware of information related to their current task.

However, in a distributed setting software engineers no longer share a physical work environment and as a consequence cannot exchange information without technological support. So, in order to collaborate with their colleagues engineers need to use technological solutions to be able to retrieve information relevant to their current task. The software engineering community has developed several solutions to fulfill this need, but most of these solutions only support a specific type of information and this information cannot be processed by other solutions directly [PR12]. Therefore, software engineers need to manually analyze the available information to be able to construct the information they need. This increased complexity of information analysis may result in misunderstandings, inconsistencies, incompatibilities and duplicated information [PR12].

To be able to acquire awareness in a relatively passive and unobtrusive fashion, such as in the co-located setting, we need to automate this analytical process of accessing, combining and filtering the available information. In essence we need to automate the process of restricting the available information to the information that an actor needs to carry out his current activity. We propose to call this mechanism a *'Virtual Office Wall'* and define it as: *"A mechanism which regulates information based on the context it encloses"*. In order to construct such an mechanism two prerequisites should be fulfilled:

(i) Access to a data set which at least contains the required data at a certain time

(ii) A method to differentiate between required and not required information

When these prerequisites are met it is straightforward how the mechanism of a virtual office wall can be constructed.

## 2.3 Selecting and Combining Information

The first prerequisite of a virtual office wall concerns having access to a data set which at least contains the required data at a certain time. In this section we discuss that to fulfill this prerequisite both access to data from a wide variety of tools is needed and the means to integrate this data to create valuable information.

In a co-located setting all information is available in a single place and software engineers are able to gather all required information in a relatively passive and unobtrusive fashion. In a distributed setting, however, all required information is scattered across multiple sites and technological solutions are needed to exchange this information. It is even impossible to collaborate effectively without some kind of technological support when people do not share a physical work environment. Therefore, in order to collaborate effectively with distributed colleagues, the software engineering community has developed a wide variety of tools. Several of these tools are widely adopted by global software engineering teams. Examples are: configuration management systems, bug trackers and Instant Messaging solutions. However, the majority of these technologies only support a single aspect of the development process. So to be able to provide the engineers with sufficient information during the entire development process many specialized tools are needed [PR12]. Because the majority of these solutions focuses on managing a specific type of information, this information cannot be processed by other solutions directly. As a consequence engineers need to manually analyze, filter and combine the available information to acquire the information they need to perform their current task. Therefore, access to a wide variety of tools is needed to fulfill the prerequisite of having access to a data set which at least contains the required data at a certain time. Because of the wide variety of systems a wide variety of access mechanisms is needed as well. Additionally, data from the different systems often needs to be combined to create valuable information. The process of combining information from different sources is often referred to as integration and we will further illustrate its value, origins and future by discussing the Coordination Pyramid defined by Sarma et al. [Sar10].

Sarma et al. [Sar10] have reviewed several software tools which assist software engineers in coordinating their efforts and proposed a framework to categorize these (see figure 2.1). This framework organizes types of existing and emerging tools in a hierarchy of paradigms of coordination shifts (the vertical axis of the framework) that have historically emerged. These paradigms are categorized along three strands: communication, artifact management and task management. These three strands represent the basic coordination activities in software development. Software engineers need to (i) communicate with each other, (ii) coordinate

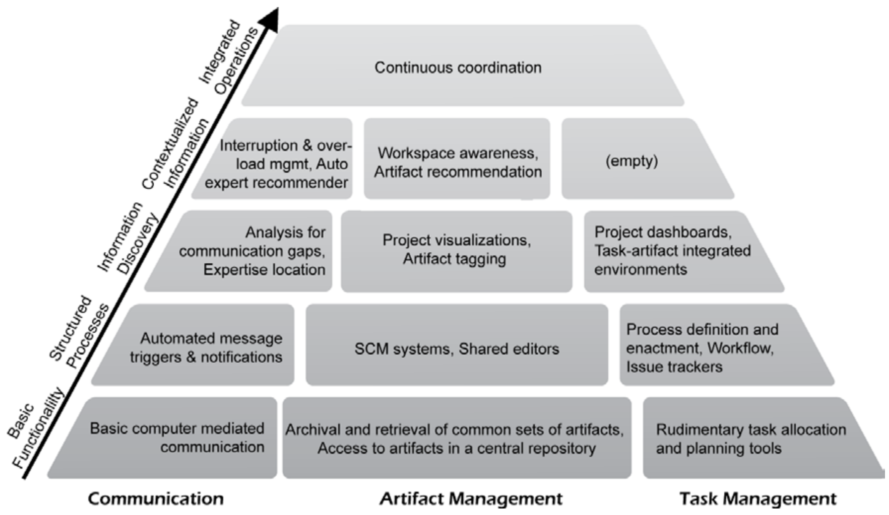

Basic Functionality
Structured Processes
Information Discovery
Contextualized Information
Integrated Operations
Continuous coordination
Interruption & overload mgmt, Auto expert recommender
Workspace awareness, Artifact recommendation
(empty)
Analysis for communication gaps, Expertise location
Project visualizations, Artifact tagging
Project dashboards, Task-artifact integrated environments
Automated message triggers & notifications
SCM systems, Shared editors
Process definition and enactment, Workflow, Issue trackers
Basic computer mediated communication
Archival and retrieval of common sets of artifacts, Access to artifacts in a central repository
Rudimentary task allocation and planning tools
Communication
Artifact Management
Task Management

**Figure 2.1:** *The Coordination Pyramid [Sar10]*

their individual access and changes to a common set of interdependent artifacts and (iii) manage their tasks. Now we briefly discuss and summarize each of the five paradigms.

The first layer in the Coordination Pyramid is the *'Basic Functionality'* layer which focuses on enabling computerized coordination. Technology at this layer allows a team to move from purely manual coordination strategies to strategies that involve automated tools. These tools, however, only focus on a specific aspect of coordination and only automate the minimal functionality needed to support this. Examples of tools in this first layer are: email, scheduling tools and shared file systems. When using such tools software engineers still make (time-consuming) decisions such as when to coordinate and with whom.

The second layer, *'Structured Processes'*, focuses on guiding the engineers in their engagement with the product and their team members. The underlying goal of tools from this layer is to enforce a specific procedure for editing, managing and relating changes to the different project artifacts. Examples of technological support that can be mapped to this layer include shared editors, issue trackers and work flow systems. These tools all reduce the coordination effort per software engineer because many coordination decisions, which take effort, are now captured by these tools. However, it can be time-consuming to explicitly model and set up the desired processes.

Subsequently, the *'Information Discovery'* layer aims to support informal practices of coordination. Informal coordination relies on users gaining information that establishes a context in which they perform their individual tasks. Tools at this layer try to provide the users with the information necessary to build this context. Examples of these tools are project dashboards, visualization systems and tool support for finding expertise. Tools from this layer combine and visualize information that is already specified by engineers as part of other tasks (e.g. commit logs, personal information, work item status) in order to automate tasks that otherwise have to be performed manually. In this layer the benefits of integrating the information from different information sources becomes clear. By combining, for example, information about artifacts that usually are modified together and information about who most frequently modified the related source code files, it becomes possible for a software engineer to pro-actively determine who best to contact in case of doubt.

Fourthly, the *'Contextualized Information'* layer, tools at this level focus on automatically predicting and providing useful coordination information to create a context in which only relevant information is exchanged in a relatively unobtrusive manner. An example of such a technology is a workspace awareness tool, such a tool provides its users with information about potential conflicting activities undertaken by other users of the system. In this layer it is essential to focus on the interplay of awareness cues presented by the tools and the responses of the engineers to these cues to be able to provide a stronger context of one's activities. Because, the stronger a context for one's activities the stronger the opportunity for engineers to self-coordinate with their colleagues to swiftly resolve any emerging coordination problems.

Finally, Sarma et al. [Sar10] leave the top of the pyramid open as they believe new paradigms of coordination will emerge as technology and organization practices continue to evolve. However, they do define the ultimate goal (the top of the pyramid) of coordination technologies: to achieve continuous coordination. In other words, the goal is to achieve *"flexible work practices supported by tools that continuously adapt their behavior and functionality so coordination problems are minimized in number and impact"* [Red07]. In this scenario software engineers no longer need to use specific coordination tools since coordination and work activities are integrated in a single environment providing its users with all the necessary information. We completely agree with this, since when all necessary information is integrated into a single environment and such an environment provides the necessary information and functionality in a seamless and effective manner it can be used to collaborate effectively with your distributed colleagues.

## 2.4 Context of an Actor

The second prerequisite of a virtual office wall concerns a method to differentiate between required and not required information. In this section we argue that a valid representation of the context of an actor is sufficient to achieve this. Therefore we introduce Activity Theory as a means to represent the context of an actor and argue it is an appropriate representation of software engineering activities as well.

Tell and Babar [Tel12] propose the use of Activity Theory in order to both structure and describe the context in which distributed software engineers perform their tasks. The origins of Activity Theory are threefold: (i) classical German philosophy, (ii) the writings of Marx and Engels and (iii) the Soviet Russian cultural-historical psychology of Vygotzky, Leont'ev and Luria [Eng99]. The theory was further improved by Leont'ev [Leo78] and became popular after Engeström introduced it to the western world. One of Engeström's main contributions is a systematic representation of the theory; the activity system (see figure 2.2).

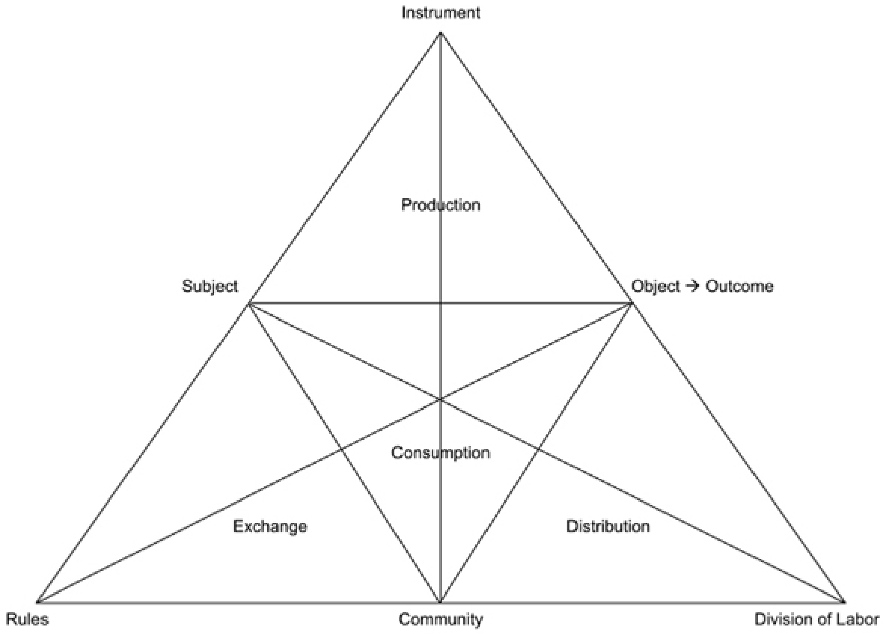


**Figure 2.2:** *Activity System [Eng87]*

This model consists of six elements:

| | |
|---|---|
| **Object** | The objective of the activity |
| **Subject** | The actor engaged in the activity (either an individual or a group) |
| **Community** | The social context of the activity (all actors involved in the activity system) |
| **Instrument** | The artifacts or concepts used by the subject of the activity |
| **Division of Labor** | The hierarchical structure of the activity |
| **Rules** | The laws, rules and regulations that govern the subject inside a community |

All these elements together represent a single human activity resulting in a single outcome.

In addition to modeling a human activity, it is also necessary to describe the hierarchical structure of that activity. Because, such a structured overview is needed to be able to relate a single human activity to activities carried out by the rest of the team. Leont'ev defined a hierarchical model of human activity which consists of three levels [Leo78]:

| | |
|---|---|
| **Activity** | is driven by its motive (e.g. a man participates in a communal hunt because he wants to feed his family) |
| **Action** | is driven by its goal (e.g. a man scares away the prey from himself and toward the other members of the hunt) |
| **Operation** | is driven by its conditions (e.g. how the man carries out the various tasks involved in his role will depend upon the weather, the terrain etc.) |

In [Tel12] Tell and Babar discuss how to use Activity Theory to describe different activities and processes of developing software in the context of GSE. They do this by describing, detailing and decomposing software engineering activities and map these to the activity system and the hierarchical model of human activity. In this discussion they use the software architecture design process as leading example. They show how this activity can be mapped to the activity system and explain each of the six elements and the outcome of this model. Subsequently, they decompose the software architecture design activity into its composing actions; architecture analysis, architecture synthesis and architecture evaluation. Next, they change the subject to the evaluation manager and emphasize on his/her motive to evaluate a candidate solution. For this activity, from the perspective of the evaluation manager, the same two mappings are applied and discussed. This decomposition can be further applied to a point at which the activity is performed through actions facilitated by technology [Tel12].

Finally, Tell and Babar [Tel12] conclude that the introduction of Activity Theory into the field of global software engineering makes it possible to determine what information is needed when performing a specific collaborative task. Because, on the one hand, applying the activity system to software engineering activities results in a uniform and detailed description of the current activity. This can be used to determine what information is directly related to the activity. On the other hand applying the hierarchical model of human activity to software engineering activities, results in an overview of the hierarchical structure of that activity. This can be used to determine the degree of relatedness between activities. In our opinion using Activity Theory is an appropriate way to describe the context of an actor and as such also to determine which information is required while performing an activity.

## 2.5 Industrial Evaluation

In the previous section we concluded that contexts described by Activity Theory are an appropriate way to represent activities of software engineers. To be able to use such contexts to differentiate between required and not required information, the contexts of actors performing the different activities which are common in software engineering need to be sufficiently different. To determine whether this is true in practice we have performed an industrial evaluation which is discussed in this section. We conducted a focus group in which we attempted to answer the following three questions:

1. Which activities are carried out most frequently?
2. Which instruments are used while carrying out an activity?
3. Which actors are involved while carrying out an activity?

### 2.5.1 Site

Participants in this study are a group of software engineers at IHomer, a Dutch software engineering company founded in August of 2008. The company is fully distributed (see figure 2.3), since the default location from which the employees work is their home. As a consequence, all employees are experienced with dealing with the difficulties of developing software when working physically separated from each other. This makes this company a suitable setting to conduct this evaluation.

### 2.5.2 Data Gathering and Analysis

To answer the three questions, we performed a focus group [Kon04, Bai78] to gather the qualitative data we needed. We conducted a focus group to gather the insights, ideas, viewpoints and opinions of the people participating because such

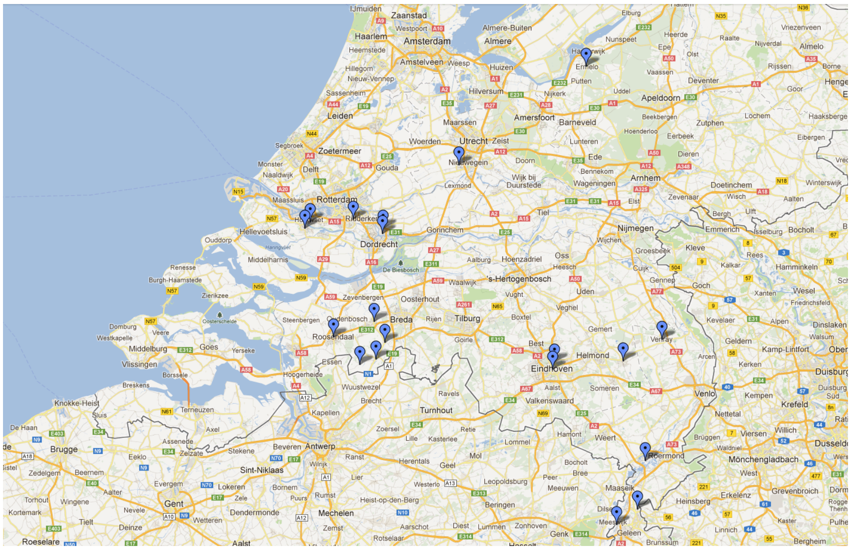


**Figure 2.3:** *Geographical distribution of the employees of IHomer*

a setting enables the participants to build on the responses and ideas of others. This process increases the richness of the information gained [Lan03]. We also used this method because of its ability to discover new insights and because it is a cost efficient way of obtaining practitioner experience.

The focus group we conducted lasted approximately 45 minutes and we selected six employees of IHomer based on their availability. The focus group itself was carried out in a separate office to minimize the influences from outside. During the focus group the author of this dissertation took the role of moderator. As a moderator he (i) explained what a focus group entails, (ii) explained what was expected of them, (iii) explained the goal of the focus group, (iv) kept the discussions on topic, (v) tried to ensure that all participants contributed to the discussion, and (vi) made sure that the predefined structure of the focus group was followed. The identification of the most frequently carried out activities was performed as follows: Firstly, the moderator handed out sticky notes and asked each of the participants to write down what they thought were the five most frequently carried out activities. Following this the moderator gathered the sticky notes from all individuals and discussed each of them with the entire group. In the discussion of each sticky note the group determined what was meant by it and grouped it together with other notes when appropriate, trying to create an overall group consensus. This process eventually resulted in the most frequently carried out activities. Finally, for each of these activities the moderator asked both which instruments are used and which actors are involved while carrying out that activity. We reached consensus in a similar fashion as with the activities.

### 2.5.3 Findings

The four most frequently carried out activities are shown in figure 2.4. We only show the four most frequently carried out activities because the participants of the focus group unanimously decided those are the most important ones. Next, we discuss the differences between the contexts of the four most common activities and reflect on these differences.

Firstly, we can see that in the four most common activities we already identified three different communities, namely: (i) team, (ii) team and organization, and (iii) team, organization and customer. These differences in actors involved in the activity make it possible to regulate information based on the social context. So, for example, when testing a new or existing piece of software engineers do not need to share this information with all their colleagues and the customer, only direct team members need to be aware of their activities.

Secondly, we can also see that the artifacts and concepts used by the subject of the activity differ between the four activities. This makes it possible to filter the information based on the current activity of an actor. When creating new software, for example, an actor does not need information about the upcoming appointments of his colleagues. Both results confirm that applying Activity Theory to the field of GSE can help provide distributed software engineers with relevant information.

### 2.5.4 Limitations

The first limitation is that we conducted one focus group with participants working for a single software engineering company. Therefore we cannot guarantee the completeness of the instruments used and the actors involved for each of the identified activities. So, to be able to draw more externally valid conclusion, the study should be repeated with a sample which more accurately represents the total population of software engineers.

There are also focus group specific limitations which have to do with group dynamics, communication styles and the social acceptability of certain topics and opinions which can all influence the discussion and therefore introduce bias [Kon04, Bai78]. We dealt with these limitations by defining and following a predefined structure to be able to control the overall content of the focus group and to make sure that group dynamics did not steer the discussion in an undesirable direction. In order to minimize the negative effects of social acceptability we emphasized the importance that everyone should contribute to the discussion. We also used sticky notes to force everyone to think about the question in advance to reduce the temptation to agree with the loudest person or the first person to give his opinion.

The final limitation is the possibility that participants have hidden agendas [Kon04]. However, it is not likely this threat to validity applies to this case study because the participants had no logical interest in influencing its outcome.

| **Activity 1: Coordination of tasks within the development team to be able to collaborate effectively** | |
|---|---|
| *Community* | *Instruments* |
| Team | Communication technologies |
| Organization | Face-to-Face meetings |
| | Documentation |
| | Issue Management System |
| | Software Repository |
| | Agenda |

| **Activity 2: Coordination of tasks between the development team and the customer to be able to collaborate effectively** | |
|---|---|
| *Community* | *Instruments* |
| Team | Communication technologies |
| Organization | Face-to-Face meetings |
| Customer | Documentation |
| | Issue Management System |
| | Software Repository |

| **Activity 3: Creation of new software to be able to add new functionality to the system** | |
|---|---|
| *Community* | *Instruments* |
| Team | Communication technologies |
| | Face-to-Face meetings |
| | Issue Management System |
| | Software Repository |
| | Development Environment |
| | Requirement Management System |
| | Testing Framework |

| **Activity 4: Testing of new or existing software to be able to guarantee the quality** | |
|---|---|
| *Community* | *Instruments* |
| Team | Communication technologies |
| | Face-to-Face meetings |
| | Documentation |
| | Development Environment |

**Figure 2.4:** *The four most frequently carried out activities*[1]

[1]Because we conducted a single focus group we cannot guarantee the completeness of the communities involved and the instruments used for each activity. Therefore, this focus group should be repeated in other settings. It is, for example, likely that an issue management system is added to the list of instruments used during testing of software (Activity 4).

## 2.6 Concluding Remarks

### 2.6.1 Conclusions

In this chapter we first compared the co-located setting with the distributed setting and concluded that in a co-located setting awareness is spread relatively passively and unobtrusively while it takes more effort in a distributed setting. In a distributed setting this information cannot be exchanged without technological support and most of these solutions only support a specific type of information. Therefore software engineers have to manually analyze the available information to construct the information they need. This increased complexity of information analysis needs to be automated to be able to acquire awareness in a relatively passive and unobtrusive fashion, such as in the co-located setting. We proposed to call such a mechanism which regulates information based on the current activity of an engineer a *'Virtual Office Wall'*. Subsequently, we discussed the two prerequisites which should be fulfilled to construct virtual office walls. The first prerequisite concerns having access to a data set which at least contains the required data at a certain time. To fulfill this prerequisite both access to data from a wide variety of tools is needed and the means to integrate this data to create valuable information. The second prerequisite of a virtual office wall concerns a method to differentiate between required and not required information. To fulfill this prerequisite a valid representation of the context of an actor is sufficient. Finally, we validated our representation of the context of a software engineer in an industrial setting and concluded that automatically erecting virtual office walls have the potential to provide global software engineers with the context of their current work activity. The main contributions of this chapter are the following:

- The definition of a virtual office wall as *"A mechanism which regulates information based on the context it encloses"*
- The prerequisite of a virtual office wall that both access to data from a wide variety of tools is needed and the means to integrate this to create valuable information
- The prerequisite of a virtual office wall that a method is needed to differentiate between required and not required information
- The applicability of Activity Theory as a way to represent the context of a software engineer
- The validity of the applicability of Activity Theory in software engineering in an industrial setting

### 2.6.2 Future Work

The next step in our research is to apply the concept of virtual office walls. We are planning to apply this concept on Iris; a cross-platform, web-based, extensible communication framework we are working on. With this platform we aim to both support awareness needs of software engineers in a single solution and enable integration of information from different sources. Currently, we are designing a way to incorporate the concept of virtual office walls in this platform to provide distributed software engineers with relevant information related to their current context. We propose to do this by providing the users with a mechanism to contextualize the available information. An example of such a mechanism is that a user can define a project group, in which all information about a specific project can be clustered, such as project members, project description, project message board, outstanding project issues and related conversations. Subsequently, the context of a user is confined to one of these projects based on his current activity. One of the main challenges in implementing this functionality is visualizing the contextualization of the information. Next to designing and implementing this concept, it should also be evaluated in an industrial setting. In this setting physically distributed collaboration should be common and the people should have experience with dealing with the difficulties this way of working entails.

### 2.6.3 Virtual Office Implications

In this chapter we have discussed how auto-erected virtual office walls can help distributed software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness. This discussion partly answered the first research question of this dissertation:

**Research Question 1**
*What are the requirements for technological support to provide distributed software engineers with the context of their current work activity?*

The requirements of a virtual office, which we can derive from the research conducted in this chapter, are:

**Req 1. Facilitate acquiring awareness relatively passively and unobtrusively**
*Software engineers should be able to acquire a sufficient level of awareness for their work activities in a relatively passive and unobtrusive fashion, similar to a co-located setting. Therefore the analytical process of accessing, combining and filtering information should be supported to avoid misunderstandings, inconsistencies, incompatibilities and duplicated information.*

**Req 2. Facilitate having access to a data set which at least contains the required data to carry out work activities**
*Software engineers need a wide variety of information to carry out their work activities. For example information about the requirements and the software repository.*

**Req 3. Facilitate combining data from different sources**
*Software engineers need to combine and integrate information from different sources to create valuable information. For example combining information about the artifacts that are usually modified together and information who most frequently modified the related source code files, makes it easier for a engineer to determine who best to contact in case of doubt.*

**Req 4. Facilitate differentiating between required and not required information**
*Software engineers should be able to differentiate between required and not required information to regulate the available information to that information they need to carry out their current work activity.*

**Req 5. Facilitate a valid representation of the context of a software engineer**
*Software engineers need a way to represent their context to determine which information is required while performing a work activity.*

Our vision on how best to support global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness only partially answered the first research question. In the next section we discuss a feasible approach to develop, implement and validate technological solutions which fulfill our vision.

Chapter 3

# An Approach for Constructing a Virtual Office

*Software engineering is increasingly carried out in distributed settings. Software engineers are becoming more nomadic in carrying out their work; working from the customer location, the headquarters of their own company, their home, or sometimes even from their holiday locations. Therefore technological support is needed to overcome the negative impacts of distance that are introduced by this trend. In this chapter we present the context in which we are bootstrapping a custom fit environment to support a team of fully dislocated software engineers and the incremental process we use. By working in this fashion we are discovering the requirements to support fully distributed teams while at the same time providing our setting with working solutions to help them with their day to day challenges. Finally, this continuous practical use also provides us with empirical data to validate the increase in awareness levels of dislocated software engineers and helps us in pinpointing important open research challenges.*

---

This chapter is strongly based on our publication *"Supporting Distributed Software Engineering in a Fully Distributed Organization"* in the Proceedings of the $5^{th}$ International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2012). This publication is co-authored by Kevin Dullemond and Rini van Solingen.

## 3.1 Introduction

In this chapter we present our approach to developing and validating technological support for distributed software engineers. The main objective of this chapter is not so much on presenting the solutions themselves, but mostly on explaining the setting in which we are bootstrapping a solution that fits a fully distributed setting, and the process we use for doing so. The angle we take in this process focuses on aiding distributed software engineers to acquire a sufficient level of awareness independent on whether they are co-located or dislocated. With awareness we mean *'an understanding of the activities of others, which provides a context for your own activities'* [Dou92]. Having sufficient awareness is essential because software engineering is an inherently cooperative activity which requires potentially many software engineers to coordinate their efforts to produce a system. In order to do this there exists a need for a shared understanding both about the project itself, like its state and its artifacts, and about the people who work on the project, like their activities, availability and interactions. Acquiring and sustaining this shared understanding is far harder in a distributed setting than in a co-located setting where it is shared relatively passively and unobtrusively [Fog05, Sch02]. This is why we focus on facilitating this relatively passively and unobtrusively in a distributed setting as well.

We approach this problem by creating a cross-platform, web-based, extensible communication framework which we co-develop with a Dutch software development company and roll out in this setting as well. The company has the policy for its software engineers to work from home as much as possible and therefore is ideal for identifying the problems faced when working distributed from your colleagues as well as verifying the viability of the solutions produced. We build this platform in two week iterations at the start of which we decide what to build based on what the users find most valuable and at the end of which we perform an evaluation. From the first iteration onwards all the engineers use the system in their daily work which enables us to acquire feedback quickly. Working in this fashion provides for a potentially short turnaround time for research subjects as identification, implementation and evaluation can be as quick as a single iteration.

The rest of this chapter is structured as follows. In section 3.2 we discuss the growing popularity of distributed teams and how collaborating in such teams is more difficult than collaborating co-located. Next, in section 3.3 we discuss which awareness needs are reasonably well supported by existing solutions and which are not. Following this, in section 3.4, we elaborate on the approach we take in this research by discussing the objectives and the reasoning behind these in 3.4.1, the setting in 3.4.2, the process we use in 3.4.3 and the implementation of the system in 3.4.4. Finally we present a summary and discuss future work in section 3.5.

## 3.2 Awareness in Distributed Software Engineering Teams

Collaborative software engineering is increasingly conducted outside of the traditional single office building for instance in multiple dislocated office buildings or from home. This is the result of the increasing globalization of business [Car99, Her01, Her07] and the rising popularity of working from home [Die09]. Advantages of the globalization of business include: market-proximity [Gri99, Dam06], reducing time-to-market by working around the clock [Car99, Ebe01], flexibility with respect to business opportunities [Car99, Her99], reducing costs by delegating work to countries with low labor cost [Car01, Dam06] and being able to fully utilize available resources [Her01, Dam06]. Advantages of working from home include: increased autonomy [Har02], increased flexibility [Har02], increased productivity [Hes91], increased motivation [Pra93] and improvement in the quality of the environment [Har02].

In collaborative software engineering, having access to the knowledge about the context in which one is working (commonly referred to as *'awareness'* [Sch02, Dou92]) is essential to properly collaborate with others [Sch02, Syr97]. Ko et al. [Ko07] reported the most frequently sought information by software engineers includes awareness about tasks, artifacts and co-workers. For example, their results show that software engineers frequently seek information about changes in artifacts they depend on, the activities of their team members and information relevant to their current task. In general there are three strategies to keep people aware of important information during collaboration: polling, alerts and peripheral awareness [Cad01]. Making the information available by polling involves making the information accessible and allowing people to explicitly poll this information on an as needed basis. Using alerts involves intentionally interrupting people to provide information. The main benefit of using alerts is that the recipient can be sure he is notified in time of important information while the main disadvantage is that it can disrupt the recipient from his current task. The final strategy to make information available is peripheral awareness. This involves making information available in the recipient's periphery such that he has access to the information without it distracting him.

In co-located teams all three of these strategies are used. Important examples of methods of sharing awareness in such a setting are meeting in the hallway [Cur88], watching other software engineers carry out their task [Seg95] and observing changes being made to artifacts [Dix04]. Also, Perry et al. [Per94] reported that software engineers spent over half their time interacting with colleagues and that most of the communication is intended to maintain awareness. In distributed teams using such methods to maintain awareness is far more difficult. In their review Omoronyia et al. [Omo10] conclude that: *"overall, the literature suggests that many aspects of awareness that are disseminated as a natural by-product of co-located working are difficult to achieve in distributed working environments."* Therefore it is important that technological support is developed that supports this process so that the advantages of distributed development can be fully exploited. Omoronyia et al. also claim that: *"to leverage the advantages of both co-located and distributed development, it is important that tool support for distributed teams aims to emulate the attributes of co-location awareness."*

## 3.3 Awareness and Technological Support

In this section we will closely follow Omoronyia et al. [Omo10] because while there exist other reviews (e.g. [Sto05, Sar05, Sch02]) this is the only review of existing technological support for awareness in distributed software engineering teams which links awareness types and their support requirements and in turn cross references awareness systems with these support requirements. Omoronyia et al. choose to focus on five types of awareness that are particularly relevant to supporting group dynamics that exist during collaborative work [Gut96, Gro05] and use these to structure the analysis of existing technological support. The requirement categories are [Omo10]:

1. Workspace Awareness: *The up-to-the-minute knowledge of other participants' interactions with the shared workspace* [Gut95]

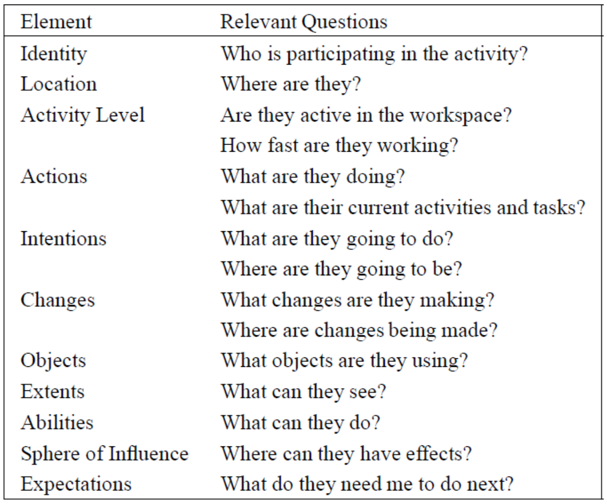
2. Informal Awareness: *The general sense of who is around, what they are doing, and what they are going to do* [Gut96]

3. Group-Structural Awareness: *Knowledge about people's roles and responsibilities, their positions on an issue, their status, and group processes* [Gut96]

4. Social Awareness: *Information about the presence and activities of people in a shared environment* [Pri99]

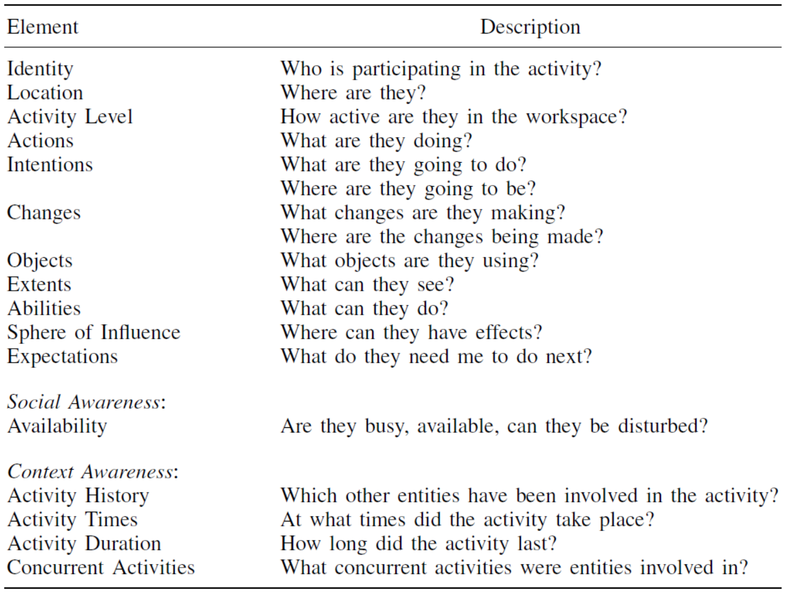
5. Context Awareness: *The evolving internal and external state information that fully characterizes the situation of each entity in a shared environment* [Omo10]

Omoronyia et al. analyze how well current solutions support awareness by comparing a set of these solutions with a list of awareness elements these can support [Omo10]. This list of awareness elements is created by extending a list of awareness elements defined by Gutwin et al. [Gut96] (see table 3.1) with additional elements based on the specific five awareness types they consider (see table 3.2 for this list). The selection of tools to consider was arrived at by going over a range of techniques that have been used to enhance awareness during distributed software development within IDEs and related tools. The techniques they considered were: social tagging, mining relationships, monitoring interactions, a combination of mining relationships and monitoring interactions and, finally, including the notion of time with that combination. The resulting summarizing table depicting which awareness elements the different tools that are discussed support is shown in table 3.3.



| Element | Relevant Questions |
|---|---|
| Identity | Who is participating in the activity? |
| Location | Where are they? |
| Activity Level | Are they active in the workspace? |
| | How fast are they working? |
| Actions | What are they doing? |
| | What are their current activities and tasks? |
| Intentions | What are they going to do? |
| | Where are they going to be? |
| Changes | What changes are they making? |
| | Where are changes being made? |
| Objects | What objects are they using? |
| Extents | What can they see? |
| Abilities | What can they do? |
| Sphere of Influence | Where can they have effects? |
| Expectations | What do they need me to do next? |

**Table 3.1:** *Elements of workspace awareness [Gut96]*



| Element | Description |
|---|---|
| Identity | Who is participating in the activity? |
| Location | Where are they? |
| Activity Level | How active are they in the workspace? |
| Actions | What are they doing? |
| Intentions | What are they going to do? |
| | Where are they going to be? |
| Changes | What changes are they making? |
| | Where are the changes being made? |
| Objects | What objects are they using? |
| Extents | What can they see? |
| Abilities | What can they do? |
| Sphere of Influence | Where can they have effects? |
| Expectations | What do they need me to do next? |
| *Social Awareness*: | |
| Availability | Are they busy, available, can they be disturbed? |
| *Context Awareness*: | |
| Activity History | Which other entities have been involved in the activity? |
| Activity Times | At what times did the activity take place? |
| Activity Duration | How long did the activity last? |
| Concurrent Activities | What concurrent activities were entities involved in? |

**Table 3.2:** *Extension of Workspace Elements to Include Social, Informal, Group-Structural and Context Awareness[Omo10]*

| Element | Description | TagSEA | Rational Team Concert/Jazz | Sisyphus | Hipikat | Expertise Browser | Palantír | FAST Dash | Team Tracks | CASS | Augur | Ariadne | Mylyn | CRI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Identity | Who is participating in the activity? | X | X | X | X |  | X | X |  | X | X | X |  | X |
| Location | Where are they? |  |  |  |  |  |  | X |  |  |  |  |  |  |
| Activity Level | Are they active in the workspace? |  | X |  |  | X | X |  |  |  |  |  | X | X |
| Actions | What are they doing? |  | X |  |  |  |  | X |  |  |  |  | X | X |
| Intentions | What are they going to do? | X | X | X |  |  |  |  |  | X | X |  |  |  |
|  | Where are they going to be? | X | X | X |  |  |  |  |  | X | X |  |  |  |
| Changes | What changes are they making? | X |  | X |  | X | X | X |  |  |  |  |  |  |
|  | Where are the changes being made? | X | X |  |  |  | X | X |  | X | X |  | X | X |
| Objects | What objects are they using? | X | X |  |  | X | X | X |  | X | X |  | X | X |
| Extents | What can they see? |  |  |  |  |  | X |  |  |  |  |  | X | X |
| Abilities | What can they do? |  | X | X | X | X |  | X | X |  |  | X |  | X |
| Sphere of Influence | Where can they have effects? |  | X |  |  | X |  |  | X |  |  |  | X | X |
| Expectations | What do they need me to do? | X | X |  |  |  |  |  |  | X | X |  |  |  |
| Availability | Are they busy, available, can they be disturbed? |  | X |  |  |  |  |  |  |  |  |  |  |  |
| Activity History | Which other entities have been involved in the activity? |  |  |  |  |  |  |  |  |  |  |  |  | X |
| Activity Times | At what times did the activity take place? |  |  |  |  |  |  |  | X |  |  |  | X | X |
| Activity Duration | How long did the activity last? |  |  |  |  |  |  |  |  |  |  |  |  | X |
| Concurrent Activities | What concurrent activities were entities involved in? |  |  |  |  |  |  |  |  |  |  |  |  | X |

**Table 3.3:** *Classification of systems-based elements of awareness.[Omo10]*

The main benefit of this last table is that it helps to identify those elements that are reasonably well supported by existing solutions and techniques and the elements that are not. Looking at the table we see that most of the elements are supported by more than one of the considered tools. The main exceptions to this are the location of software engineers, the extent to which they are available and all but one of the elements related to context awareness. Therefore these seem like good areas to direct further research.

## 3.4 Approach

In this section we discuss our approach to researching supporting awareness with technology. First we discuss the main objectives we want to achieve and the reasoning behind these objectives. Following this we discuss how we aim to reach these objectives by describing the setting in which the solution will be evaluated and the process we employ in developing and evaluating the solution in that setting. Finally we also briefly discuss how we are implementing the proposed system.

### 3.4.1 Objectives

The goal of our research is to determine how best to support distributed software engineering with technological support for aiding people to acquire sufficient awareness. The two core objectives of our approach to achieve this are the following:

1. Support awareness needs of software engineers in a single platform
2. Enable integration of the information from different sources

We arrived at these objectives as follows. Having researched the extensive literature of existing attempts at supporting awareness for distributed software engineers we identified a pitfall many of these approaches suffer from. Most tools are designed to help resolve a specific type of question and target a specific software artifact. Sillito [Sil08] reports on an empirical study on how programmers resolve change tasks and how tools support them in answering questions they have in the process of carrying out these tasks. He reports that most of the tools that he researched treat questions as if they are asked in isolation while they often are, in fact, part of a larger process. Examples are asking questions on different levels of abstraction and asking questions involving different information sources. Because awareness questions are often part of a larger process, involving a series of questions and activities that provide context, it is often difficult for distributed software engineers to obtain sufficient contextual awareness [Omo10]. Therefore we feel it is highly valuable for a tool supporting awareness for distributed software engineers to facilitate combining the different types of information. Firstly, we propose to do this by providing a single platform to support awareness needs of distributed software engineers. Secondly, Sillito [Sil08] also states that even programs that do support asking different questions generally fail at combining the information in a useful way and merely report the information in isolation as largely undifferentiated and unconnected lists. Therefore we also think it is important to enable the integration of information from different sources.

### 3.4.2 Setting

The development and evaluation of the solution we are creating is relatively unique because it is done at a company called IHomer, a Dutch software engineering company founded in August of 2008, which is distributed in the true sense of the word. The company employs highly responsible, proactive people referred to as participants (instead of employees). All participants are responsible for all business decisions like the strategy, vision and core values, in contrast with employees at 'regular' companies who are mainly responsible for the specific role they fulfill.

In the company, physically distributed collaboration is common since participants aim to work from home as much as possible. This makes the company a particularly suitable setting for performing this research because of two reasons. Firstly, the people are quite experienced with dealing with the difficulties of working distributed from each other and therefore have quite a good understanding of what is needed to improve this situation. Because of this we can closely collaborate with the other participants to determine which types of awareness are most beneficial to support first, and what good ways to achieve this are. Next to this, the other participants also collaborate with us in realizing the actual technical implementations which improves the quality of the solutions and reduces the time it takes to realize these. Secondly, we can also perform high quality evaluations because the company perfectly matches the target setting for which we are attempting to solve issues: a fully distributed organization. These evaluations can also be done in a lightweight manner and with low turnaround time because of the high quality feedback the other participants can give us due to their experience with distributed collaboration. Finally, because the participants encounter the issues we are attempting to solve in their daily work the solutions will also benefit them directly.

### 3.4.3 Process

Based on the specific characteristics of the project, the setting we are conducting research in and our own experiences in the past, the process we use should fulfill three requirements. Firstly, it should be able to cope with uncertainty and changing requirements since we are creating a genuinely novel product and projects creating genuinely novel products are often faced with uncertainty regarding both requirements and implementation technologies. Secondly, it should involve the intended users of the system as strongly as possible because they are quite experienced with working in a distributed setting since this is something they encounter on a daily basis. Finally, it should stimulate the usage of the platform by all users by providing value as soon as possible. This is important because we found that the value of awareness sharing technology (CSCW groupware) is higher when a larger portion of the team uses it, see chapter 6, and that this is often a problem when introducing such tools in practical settings.

We elect to use an agile process methodology to realize the platform we are creating because such a methodology fulfills all three of these requirements. Firstly, such a methodology is better able to cope with uncertainty and changing requirements in projects than plan-driven approaches [Dul09]. The main ways in which this is accomplished is by acquiring rapid feedback from the actual users of the system by using short iterations, rapid deployment and working closely with the customers. Working closely with the customers is done to acquire feedback but also to discover how value can be created as quickly as possible resulting in increased customer satisfaction and commitment. The specific methodology we elect to use is Scrum [Sch11, Sch95] an agile process which emphasizes a set of project management values and practices [Lar04]. It does not define any specific software development techniques for the implementation phase but concentrates on how the team members should function in order to produce the system in a flexible way in a constantly changing environment [Abr02].

In the specific way we have implemented the Scrum process, we work with two-week iterations, referred to as sprints in Scrum. An overview of how our sprint looks like is depicted in figure 3.1 Each sprint starts with a sprint planning meeting in which we decide what to do in the sprint based on the product backlog, which is a prioritized list of features for the product, and an estimate of the amount of work of the different user stories on the product backlog: we decide on the sprint backlog. At the end of each sprint we perform a sprint review and a sprint retrospective. The goal of the sprint review is to discuss what has been done in the sprint and compare this to what was agreed upon in the sprint planning meeting at the start of the sprint. The sprint review revolves around reviewing the product and also includes a demonstration of the product. The sprint retrospective revolves around reviewing the process and is intended improve this in the next sprint.

For practical reasons we hold the sprint review and sprint retrospective of one sprint on the same day as the sprint planning meeting of the next sprint. We do this because we require the same group of people to be present at all three meetings, namely the product owner, the development team and a specific unchanging subgroup of the stakeholders. The stakeholders are all the participants at IHomer because these are all users of the system we are creating. The product owner is one of them, and he is responsible for representing all stakeholders and making decisions based on this responsibility. We have decided to include a subgroup of other stakeholders, next to the product owner, in the sprint review and sprint planning meeting to make it easier for the product owner to determine the

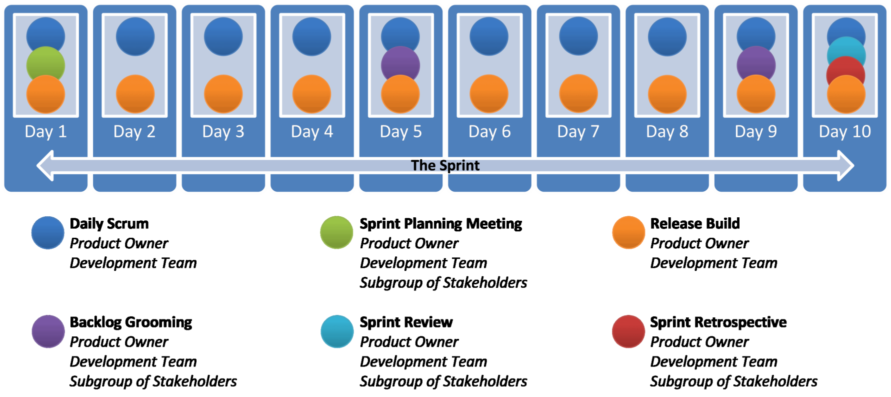


**Figure 3.1:** *Sprint Overview*

point of view of the other stakeholders and make decisions based on this. Further, we have included the subgroup of stakeholders in the sprint retrospective as well because in the way we implemented the process, with continual deployment and direct feedback from the stakeholders, the stakeholders are an important and direct part of the process and aspects of the process which include them should be analyzed and improved upon as well.

During a sprint we perform a daily Scrum and release a new build every day. The daily Scrum is a 15-minute time-boxed event for the development team and the product owner to synchronize activities and create a plan for the next 24 hours. This is done by inspecting the work since the last daily scrum and forecasting the work that could be done before the next one. We release a new built every day to provide value and acquire feedback as soon as possible. Two times during a sprint we perform backlog grooming, once on the half-way point and once at the end, on the day before the sprint review, retrospective and planning meeting. Backlog grooming is the act of adding detail, estimates, and order to items on the product backlog. The final role in our process is that of the Scrum master, who is responsible for ensuring the process is followed and impediments are removed. This role is performed by the different members of the development team on a rotation basis.

To provide all stakeholders of the project with real time insight in the project and the progress, we work with a Scrum-board which we maintain and share using Trello[1]. This is a collaboration tool which organizes projects into boards. On such a board items move along various stages of progress. We use the following stages:

- *Idea Box*: All stakeholders can insert ideas here, but also problems they encounter like actual bugs or other types of feedback. Items can be picked up and put on the product backlog when appropriate

[1]www.trello.com

- *Product Backlog*: Prioritized list of features for the product
- *Sprint backlog*: list of work items that are (going to be) implemented in the current sprint
- *Under development*: Work items that are currently being worked on
- *Deployed in current sprint*: Work items completed in the current sprint that are already deployed in the live system

Items can be inserted on various stages on the board based on their actual current status. We differentiate between six types of items on this board:

- *Feedback*: Reaction to how the system currently functions and explanation of why this is insufficient/sub-optimal
- *Idea*: A rough idea of an addition or alteration to the product backlog
- *User Story*: One or more sentences in the everyday or business language of the end user that captures something the user wants to achieve using the system
- *Defect*: Report of behavior of the system that is in contrast with how the system should function
- *Task*: A unit of work generally between four and sixteen hours which contributes to a backlog item
- *Research Task*: Because we also perform research and write papers in this project we place these on the backlog as well to increase transparency and facilitate sprint planning.

### 3.4.4 Implementation

In section 3.4.1 we discussed the two core objectives of our approach: (i) support awareness needs of software engineers in a single platform and (ii) enable integration of the information from different sources. These objectives place some constraints on our choice of implementation technology. Firstly, the technology should be cross platform because we want to support awareness needs in a single platform and our stakeholders use a variety of systems such as Linux, Windows and Mac OS, but also a number of Mobile Operating Systems running on tablets and phones. The second constraint originating from the single platform objective is that the technology should allow for scalability. We choose to support scalability by using peer-to-peer communication between the users of the system as much as possible. Our second objective leads to a third constraint on the technology we choose, namely that this should be extensible to facilitate the integration of information from different sources. Using the Model-View-Controller design pattern results in more extensible code because the model, view and controller are split into separate, loosely coupled components which make it possible to adapt the flow of the application without changing the model or the view. Using an approach with a central event bus also leads to a more extensible solution since this makes it possible to extend an application without altering the existing solution.

## 3.5 Concluding Remarks

### 3.5.1 Summary

In this chapter we have presented the research we are currently performing to determine how best to support distributed software engineering with technological support by aiding people to acquire sufficient awareness. First we discussed this is an important topic because of the increasing popularity of distributed development and the challenges this causes. Following this we talked about existing work in facilitating distributed development by supporting the sharing of awareness with technological solutions and about what these solutions support well and where there exist possibilities for improvement. Next, we discussed an iterative approach to (i) identify real-life global software engineering problems, (ii) propose and implement solutions for these problems, and (iii) evaluate these solutions in an industrial setting. This approach provides us with empirical data to validate the increase in awareness levels of dislocated software engineers and helps us in pinpointing important open research challenges. Another advantage of using this approach is that we are discovering the requirements to support fully distributed teams, while at the same time we are providing our setting with working solutions for issues they encounter in their daily work.

The most valuable contribution of this chapter is: *The description of our own unique approach to research how to support awareness in distributed software engineering in which we collaborate with software engineers in a fully distributed company to identify the encountered problems, produce solutions using an agile process and to evaluate these solutions.*

### 3.5.2 Future Work

During the first sprints we have generated some interesting ideas for future work. It would for example be interesting to provide insight into the occurrence and content of the communication of your colleagues. This is both true when working together on a project at the same time (e.g. overhearing a conversation of colleagues) but also when catching up on what happened on the project during your absence. Another example is to let the platform help the users to select the most appropriate form to communicate with their colleagues. Users could for example configure which means of communication they favor in certain situations (e.g. when in a car) and it can also be shown when certain means of communication are currently infeasible (e.g. audio call during a meeting). Similarly the system could also aid in determining who to contact based on a specific question or problem (e.g. who could help me with this specific class?). Finally, it is also possible to show something such as the mood of people in the platform. When working in the same room it is often clear what someone his mood is and when for example someone is unhappy for a prolonged period of time, steps can be taken to find the cause and solve this. When working distributed from each other, such things can go unnoticed and lead to larger problems which can be harder to solve.

### 3.5.3 Virtual Office Implications

The approach discussed in this chapter, to develop, implement and validate solutions which support global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness, is an important part in answering the first research question of this dissertation:

**Research Question 1**

*What are the requirements for technological support to provide distributed software engineers with the context of their current work activity?*

In this research we did not identify new requirements of a virtual office itself. We could, however, derive the following requirements on how to construct such an office:

- Facilitate coping with uncertainty and changing requirements
- Facilitate involving the intended users of the system as strongly as possible
- Facilitate stimulating the usage of the platform by all users

Both our vision on how best to support global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness, and our approach to develop, implement and validate such solutions contribute to answering the first research question of this dissertation. In the remainder of this dissertation we will use this vision to answer the other research questions and to elaborate on the findings. This enables us to derive a set of requirements a virtual office should fulfill.

# Part III

# Communicating in a Virtual Office

Chapter 4

# Open Conversation Spaces

*Conversations between colleagues in collaborative software engineering are important for coordinating work, sharing knowledge and creating knowledge. Overhearing conversations of others is useful as well since this: (i) provides access to the information discussed in the conversations, (ii) offers the possibility of joining the conversations and (iii) provides insight in the communication structure of the project team. When working in a global software engineering setting, specialized tooling is required to be able to have conversations and to know what conversations others are having. In this chapter we define the concept of a conversation, discuss the reasons for having conversations and discuss the advantages of overhearing conversations. These insights led us to the definition of Open Conversation Space; a space in which conversations are possible between the actors in that space and these conversations are visible to other actors in it. Finally, we present the requirements such a space should implement.*

This chapter is based on both our publication *"Virtual Open Conversation Spaces: Towards Improved Awareness in a GSE Setting"* in the Proceedings of the 5th International Conference on Global Software Engineering (ICGSE 2010), and on our publication *"Overhearing Conversations in Global Software Engineering - Requirements and an Implementation"* in the Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2011). Both publications are co-authored by Kevin Dullemond and Rini van Solingen.

## 4.1 Introduction

Global software engineering (GSE) is becoming increasingly interesting due to the globalization of business [Car99, Her01, Dam06, Her07, Pri07, Åge08]. In GSE the software development process is distributed between several geographically dispersed locations [Con06, Dam06, San06]. Advantages of GSE include: market-proximity [Gri99, Her00, Dam06], reducing time-to-market by working around the clock [Car99, Her99, Ebe01, Dam06], flexibility with respect to business opportunities [Car99, Her99], reducing costs by delegating work to countries with low labor cost [Car01, Dam06] and being able to fully utilize available resources [Her01, Gri99, Dam06]. Besides being beneficial, GSE introduces a number of challenges in relation to communication, coordination and control of the development process [Car01]. Examples are: lack of informal communication [Car99, Her99, Her01, Åge05], reduced hours of collaboration [Bat01, Kie03, Hol06, Åge08], communication delay [Åge05, Her05, Her07, Con06], and loss of cohesion [Car99, Her03, Her07].

In collaborative work it is essential to have knowledge about the context in which you are working to properly collaborate with others [Sch02, Syr97]. With information about the context we mean information about the other members in the project team, their activities, information about the state of the project and so on. This information is essential because this knowledge is necessary for coordinating actions, managing coupling, discussing tasks, anticipating others' actions, and finding help [Sch02, Syr97, Gut02]. The complexity and interdependency of software systems (e.g., [Kra95]) suggest that this is also the case for collaborative software development. In scientific literature the term 'awareness' is often used to denote this [Sch02, Dou92]. Dourish and Bellotti use the following definition: *"An understanding of the activities of others which provides a context for your own activity"* [Dou92].

Awareness is spread among the members of the project team as follows: Actors display information on a shared medium while other actors monitor the medium and acquire information from it [Sch02, DS11]. In this process, both monitoring and displaying are activities that are not necessarily conducted with the full attention of the actor. Often when expressing this varying degree of attention dichotomies are used, such as: explicit versus implicit, deliberate versus automatic, conscious versus subconscious, focused versus unfocused and active versus passive. These notions are however false dichotomies as the distinction is not categorical but merely one of degrees [Sch02].

When team members are not sharing a physical work environment they are outside of sensory range of each other. Therefore information exchange between them becomes infeasible without some kind of technological support. This can be dealt with by providing other ways of acquiring the required information, like using the telephone or email to ask a question. However, in general, such solutions are inferior to the way contextual information is shared in a traditional co-located setting, in the sense that in comparison it (i) takes more effort because the communication is more intentional [Gut04], (ii) is more obtrusive [Fog05], (iii) happens less frequently [Her03, All77, Gal90], and (iv) contains less information [Gut04, Her99, Ols96]. As such we can conclude that spreading awareness is more difficult in a distributed setting. Due to the nature of the challenges associated with GSE, it is plausible to assume these challenges originate from having insufficient access to information regarding the work context: a lack of awareness.

The research presented in this chapter continues upon this insight that a lack of awareness is the origin of the challenges faced in GSE. It is part of the ASPIC[1] research project. The goal of ASPIC is to develop solutions to the problems caused by the difficulties with acquiring and maintaining awareness in GSE. In this research the focus will lie on making the sharing of information a more passive activity because (i) this will likely lower the effort to spread awareness, (ii) cause this information to be more recent and (iii) improve the quality of the information as well. In this chapter we will focus on how knowledge about the conversations between the members of a development team can improve collaboration in GSE. The main goal of this chapter is to find out how awareness about conversations within a development team supports collaborative software engineering.

This chapter is structured as follows: In section 4.2 we give a formal definition of a conversation within the context of GSE. In section 4.3 we discuss the advantages of conversations, the advantages of overhearing conversations and introduce the concept of an open conversation space. Subsequently, in section 4.4, we define a set of requirements such a space should implement. Finally, we conclude upon our research in section 4.5.

## 4.2 Conversations

There are many definitions of the word conversation. The Oxford English dictionary for example defines it as: *"An informal spoken exchange of news and ideas between two or more people"* [Soa03]. The Merriam-Webster's collegiate dictionary uses the following definition: *"oral exchange of sentiments, observations, opinions or ideas"* [Mis03]. Finally the Cambridge advanced learner's dictionary defines conversation as: *"(a) talk between two or more people in which thoughts, feelings and ideas are expressed, questions are asked and answered, or news and information are exchanged"* [Woo08]. These definitions seem to agree on the fact that, conversations:

1. Use verbal communication

2. Are an exchange of information of various origins between two or more people

[1]Awareness-based Support Project for Interpersonal Collaboration in Software Engineering, http://aspic.ewi.tudelft.nl

We find the confinement to verbal communication too strict however. This confinement presumably originated from how people have held conversations for a very long time, namely by being at the same place at the same time and talking to each other. We feel, however, that the confinement to verbal communication is meant to convey something else altogether. Firstly, an exchange of information between people is only a conversation when the communication can be considered synchronous[2]. Because the communication should be synchronous for an information exchange to be a conversation, a form of communication that is comprehensible by humans in real time should be used. To help distinguish it from conversations, we propose to use the term correspondence for the exchange of information between people using asynchronous communication. Secondly, when people are part of a conversation they are directing their communication at one or more specific people. So, broadcasting information, like for instance an announcer working at a train station, at a football stadium or at the market place, is not a conversation. It can however be a way to initiate a conversation. Another way to initiate a conversation is by directly starting to communicate with one or more specific people. Summarizing, we define a conversation in the context of GSE as: *"An exchange of information between two or more people where those participating use synchronous communication directed at the other participants"*.

## 4.3 Open Conversation Space

In collaborative work conversations have various uses. For one, conversations help people to integrate and coordinate their work, by discussing their past, current and future activities [Esp03, Gut02, Ren11]. An example of this is when a developer ($d_1$) tells a colleague ($d_2$) he is currently working on a certain work item and what work item he is planning to do next. Because developer $d_1$ informs developer $d_2$ of his current and planned future activities, developer $d_2$ can both adapt his own current and planned activities as well as influence those of developer $d_1$. Secondly conversations are a powerful tool to share knowledge about the actual work [Web93, Eri99]. An example of this is when a project member asks a colleague to explain some technical aspect of the work he is doing. Finally, conversations are ways of creating new knowledge [Wyn79, Web93, Eri99]. Examples of this are having a discussion with someone to come up with a solution to some issue in the project and having a brainstorm session to identify the requirements of a system to be built.

Besides taking part in conversations, also conversations of others are useful. For one, overhearing the conversations of others is a source of information since this provides access to the information which is discussed and/or concluded in the conversation [Gut02]. This refers to both technical project information and information regarding the current, past and future activities of other project mem-

[2]Communication is regarded *'synchronous'* when the sending and receipt of messages between actors communicating can be regarded as instantaneous [Dul09]

bers. Secondly, having insight in the ongoing conversations provides the opportunity to join a conversation [Gre01]. Having joined a conversation, being part of that conversation provides the same benefits as discussed earlier for conversations in general, namely: integrating collaborative activities, knowledge sharing, and the creation of new knowledge. Finally, by having access to the communication frequencies between colleagues, the insight into the communication structure of the project team is increased [Sos02, McC93, Kra90]. This information is important to be able to address communication issues [Ehr07, Cat06, Cro05]. These communication structures and issues can be very dynamic and can evolve over time. An example when this information can be useful is the following: Say you need to ask someone a question but you cannot reach him. If you know who that person frequently communicates with, you could attempt to reach this person instead and see if he can assist you with contacting the right person or with resolving the issue itself.

Having discussed the benefits of conversations in general and the benefits of having access to the conversations of others, we can conclude both being able to have and overhear conversations is important in collaborative software engineering. To refer to an environment in which this is possible, we define an *'Open Conversation Space'*: A space in which (i) conversations are possible between the actors in that space and (ii) these conversations are visible to other actors in it. An example is a normal office setting. In such a setting members of the project team converse by means of spoken natural language (among other means) and such conversations are audible by other people in the setting. In fact, being able to work in a space in which the members of the project team are frequently able to both see and hear each other is one of the main reasons for working in a co-located setting [Kra90]. Therefore we propose the creation of a *'Virtual Open Conversation Space'*: An open conversation space which is applicable in a distributed setting.

## 4.4 Requirements of an Open Conversation Space

Having introduced the concept of an open conversation space we will discuss the five requirements such a space should implement in this section. These five requirements are:

**REQ1.** Facilitate starting conversations

**REQ2.** Facilitate detecting active conversations

**REQ3.** Facilitate monitoring active conversations

**REQ4.** Facilitate participating in conversations

**REQ5.** Facilitate the finishing of conversations

We have derived these requirements by analyzing the life cycle of a conversation in a structured fashion. First the conversation is started in some way, subsequently the conversation is active for a certain amount of time and finally the conversation ends and reaches the end state of being a finished conversation. We will also use these three states to structure the discussion of the requirements of an open conversation space. In this discussion we will illustrate the concepts by showing how the requirements are implemented in co-located situations, in particular the traditional office setting.

### 4.4.1 Uninitialized Conversation

In an open conversation space the actors should be able to have a conversation and therefore it should be possible for conversations to be initiated (**REQ1**). In general, there are two ways to initiate a conversation: direct and indirect. Firstly, in an office setting the most common way to initiate a conversation is by walking up to a colleague and starting to talk to him or her. Basically, you choose a specific person, or a specific group of people to initially participate in the conversation. This participant-based kind of conversation initiation is a direct way to initiate a conversation. Namely, the initiation of the conversation is part of the conversation because the communication is synchronous and directed at the other participants. Examples outside of the co-located office are calling someone on the phone or sending an IM-message to someone. It is also possible to initiate a conversation indirectly. Examples are asking for help in general and making an announcement. In a traditional office setting this can be done by talking out loud or writing something on a white-board while outside of the traditional office setting a chat room or a forum can be used. Note that when initiating a conversation in this fashion, the initiation is *not* part of the conversation because the communication is *not* directed at the other participants (there are none), but rather at a certain group of potential participants.

### 4.4.2 Active Conversation

By definition, once a conversation is initiated, an open conversation space should allow the overhearing of this conversation by the other actors in that space (**REQ2**). Firstly, they should be able to find out about the conversation either by deliberately (manually) looking for it or by automatic detection. In a traditional office setting an example of the former is looking around actively, checking to see if people are having a conversation. An example of the latter is detecting a conversation because you hear people talk to each other or see some people group together. When looking at detecting a conversation in this fashion, it is important to note it happens mostly subconsciously and unobtrusively. When you are working on a task and people talk to each other, you can continue relatively uninterrupted while your mind automatically detects whether the conversation is interesting to you. After detection of the conversation the open conversation space should allow the actors to actively monitor the conversation (**REQ3**). In a traditional office setting this would mean actively listening to the conversation without actually joining. When actively listening to the conversation an actor has access to nearly all information about the conversation, so the things that are being said, who are participating in the conversation and who else is viewing the conversation. When not actively following a conversation, more general information about the conversation is picked up by the actor, like a phrase or a certain word.

Besides monitoring an active conversation, actors in an open conversation space should also be able to participate in such a conversation (**REQ4**). People can become a participant in a conversation (i) because they were one of the original participants in a conversation, (ii) because they were invited into a running conversation or (iii) because they actively joined a conversation they overheard. In a traditional office setting people are usually invited into the conversation by simply being asked to do so by someone already participating. Actively joining a conversation yourself can happen in multiple ways. Someone listening to the conversation can explicitly ask whether he can join, but often people will join conversations by just starting to speak. When someone participates in a conversation he can influence the conversation. He can, for example, contribute to the conversation or invite other people. Next to this he can also suggest to move the conversation to a separate office if the conversation is of a private nature, effectively taking it out of the open conversation space.

### 4.4.3 Finished Conversation

Because we define a conversation to be a synchronous information exchange, by definition, it has to finish as well since people cannot synchronously communicate indefinitely. Therefore a conversation finishes when all participants stop communicating synchronously (**REQ5**). This is however not straightforward to detect. For instance, there is no definition of a certain amount of time between messages indicating the synchronous communication has ended. In a traditional office setting participants not talking for a certain amount of time and participants physically moving away from each other are usually indicators that a certain conversation has ended. Later on, it is possible for the same people to continue *"where they left off"*. This is however not the same conversation, but a second conversation about a related subject since conversations are synchronous exchanges of information.

When a conversation is finished however it does not cease to exist. In a traditional office setting people that participated or overheard the conversation usually have a recollection of the conversation while other people present in the office during the conversation can have some knowledge about it as well. Knowing about finished conversations has all benefits of overhearing conversations except being able to join the conversation since this is no longer possible.

## 4.5 Concluding Remarks

### 4.5.1 Summary

In this chapter we have answered the following research question: *"How can awareness about conversations within a development team support collaborative software engineering?"* We answered this research question, by defining the concept of a conversation, discussing the reasons for having conversations, and discussing the advantages of overhearing the conversations of colleagues. These advantages are the following: (i) it provides access to the information discussed in the conversations, (ii) it offers the possibility of joining the conversations, and (iii) it provides insight in the communication structure of the project team. This led us to conclude that a work environment should both provide the possibility of having conversations with colleagues and make the conversations going on in the work space visible. We called such an environment an open conversation space and presented a set of requirements such a space should fulfill. Finally we proposed the creation of a virtual open conversation space; an open conversation space which is applicable in a distributed setting.

### 4.5.2 Virtual Office Implications

In this chapter we provided a theoretical motivation why the overhearing of conversations of others is valuable to a distributed software engineering team. This motivation contributes to answering the second research question of this dissertation, namely:

**Research Question 2**
*What is the value of overhearing conversations in global software engineering?*

Based on this theoretical motivation why the overhearing of conversations of others is valuable to a distributed software engineering team, we derived the following set of requirements which should be implemented in a virtual office:

**Req 6. Facilitate starting conversations**
*Software engineers should be able to have conversations, therefore it should be possible to initiate a conversation. For example by choosing a specific person, or a specific group of people to initially participate in the conversation.*

**Req 7. Facilitate detecting active conversations**
*Software engineers should be able to overhear conversations of others, therefore it should be possible to find out about active conversations. Software engineers could, for example, detect a conversation because they hear or see people talk to each other.*

**Req 8. Facilitate monitoring active conversations**
*Software engineers should be able to access information about the conversation without actually joining it.*

**Req 9. Facilitate participating in conversations**
*Software engineers should be able to become a participant in a conversation. They can, for example, become a participant in a conversation because they are invited into an ongoing conversation or because they actively joined a conversation they overheard.*

**Req 10. Facilitate finishing conversations**
*Software engineers should be able to finish a conversation.*

**Req 11. Facilitate having conversations which cannot be overheard by others**
*Software engineers should be able to have conversations of a private nature, which cannot be overheard by others.*

**Req 12. Facilitate changing the degree of involvement in a conversation**
*Software engineers should be able to change how aware they are of a conversation by changing their degree of involvement. They should, for example, be able to change their involvement from actively listening to participating in a conversation.*

**Req 13. Facilitate having insight in the finished conversations**
*Software engineers should be able to find out about a finished conversation to access information about it.*

Next, in chapter 5 we evaluate the value of the concept of overhearing conversations in the field of software engineering, and in chapter 6 we evaluate the value of overhearing conversations in the field of global software engineering from actual industrial experience. These three studies together answer the second research question of this dissertation.

Chapter 5

# Evaluating the Concept of Open Conversation Spaces

*Software engineering is by nature a highly collaborative activity and being able to collaborate effectively is a key factor for project success. However, collaborating effectively in global software engineering, in which team members are geographically, temporally and socio-culturally separated from each other, is an important challenge. In a traditional co-located software engineering setting, one of the most important communication patterns is a conversation. Technological support to have conversations in a distributed setting is commonly used, however overhearing conversations of your colleagues is mostly not feasible with these tools. To explore the importance of overhearing conversations we conducted a focus group and a questionnaire in a large international software development company. In the focus group we identify: (i) the benefits and challenges of having insight in active conversations, (ii) the important types of information about a conversation, (iii) the actions possible on a conversation, and (iv) the benefits and challenges of having access to the finished conversations. In the questionnaire among 47 software engineers we determine the relative importance of these benefits, challenges, information items and possible actions. Based on these findings we conclude that research about support for conversations in global software engineering is worth pursuing and provide valuable insights on important aspects to consider when doing so.*

---

This chapter is strongly based on our publication *"An Exploratory Study on Open Conversation Spaces in Global Software Engineering"* in the Proceedings of the $7^{th}$ International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2011). This publication is co-authored by Kevin Dullemond and Rini van Solingen

## 5.1 Introduction

In this chapter we will report on the first part of an empirical study about overhearing conversations and how to support this. This first part of the study concerns the evaluation of being able to overhear conversations of your colleagues in the field of software engineering. The second part of the study will concern the evaluation of overhearing conversations in a distributed setting and will be discussed in the next chapter. We are interested in an evaluation of the value of overhearing conversations because this is often infeasible in distributed settings and could be one of the causes of the challenges faced when working in such a setting. The objective of this chapter is:

*"To provide evidence that research about support for overhearing conversations is worth pursuing and to provide insights on important aspects to consider when doing so"*

Therefore, we:

- Determine the benefits and challenges of having insight in the active conversations and determine how important these are
- Determine what information about a conversation is important and determine how important this is
- Determine what actions can be carried out on a conversation and determine how important they are
- Determine the benefits and challenges of having access to conversations after they end and determine how important these are

The remainder of this chapter is structured as follows. In section 5.2 we present background information about global software engineering, awareness issues in such settings and the overhearing of conversations. Following this, in section 5.3, we discuss the research site and our methods of data collection and analysis. Subsequently, we present our findings in section 5.4 and reflect upon the results and discuss limitations in section 5.5. Finally we will conclude upon our work and discuss future research opportunities in section 5.6.

## 5.2 Background

It is becoming increasingly common for collaborative software engineering teams to no longer conduct their work from a single office building. This happens both due to the globalization of business [Car99, Her01, Her07] and because people are starting to work from home more and more [Die09]. Advantages of the globalization of business include: market-proximity [Gri99, Dam06], reducing time-to-market by working around the clock [Car99, Ebe01], flexibility with respect to business opportunities [Car99, Her99], reducing costs by delegating work to countries with low labor cost [Car01, Dam06] and being able to fully utilize available resources [Her01, Dam06]. Advantages of working from home include: increased autonomy [Har02], increased flexibility [Har02], increased productivity [Hes91], increased motivation [Pra93] and improvement in the quality of the environment [Har02]. Since team members do not share a physical work environment when working distributed from each other, information exchange between them becomes infeasible without technological support. This information exchange, however, is necessary to acquire knowledge about the context in which you are working. This knowledge is essential in collaborative work to properly collaborate with others [Sch02, Syr97] and is commonly referred to as *'awareness'* [Sch02, Dou92]. In general, however, the technological support used to acquire awareness (such as telephone or email) is inferior to the way contextual information is shared in a traditional co-located setting, because in comparison it (i) takes more effort since the communication is more intentional [Gut04], (ii) is more obtrusive [Fog05], (iii) happens less frequently [Her03, All77] and (iv) contains less information [Gut04, Ols96].

One of the most important communication patterns that occur in a traditional office setting are conversations [Per94]. In the previous chapter we defined a conversation in the context of global software engineering as: *"An exchange of information between two or more people where those participating use synchronous communication directed at the other participants"*. So, for example, broadcasting information, like an announcer at a football stadium, at a train station or at the market place, is not regarded as a conversation. Next, to this definition of a conversation, we also discussed the importance of conversations and the importance of being able to overhear conversations of others. Finally, we introduced the concept of an open conversation space to denote a space in which actors can have conversations and where these conversations can be overheard by the other actors in that space. We will use this notion in this chapter as well.

## 5.3 Research Site and Method

### 5.3.1 Site

Participants in the study are a group of software engineers at Exact, a software development company operating in 40 countries. Exact offers Enterprise Resource Planning software for medium-sized and small businesses. At the end of 2010 it employed 1867 employees worldwide and 359 in the Netherlands alone [Exa10]. The specific group of employees that are involved in the study worked on a product called Exact Online which is offered as a service (SaaS). Currently, this product is targeted at the lower end of the SME market and was introduced in the Netherlands in 2005. The majority of the people involved in this study (42), worked out of the office location in Delft (The Netherlands) and was co-located on a single floor. However, also three people from the Wemmel (Belgium) office participated as well as two from the Minneapolis (USA) office. Next to this, people worked from home fairly often and frequently communicated using Instant Messaging software even when working from the same office.

The group of people that participated in the study is in our opinion appropriate for reaching the research objectives described in the introduction. Firstly, the group of people consisted of experienced software engineers with an average of 10 years of experience in the field. Secondly, the group of people we studied is both experienced with collaborating in a co-located setting and in a distributed setting. Finally, the group also used Communico (a tool which makes it possible to overhear Instant Messaging conversations, see chapter 6) and therefore gained familiarity with the concepts researched in the study.

### 5.3.2 Data Collection and Analysis Methods

To reach our research objectives we used two methods to acquire the empirical data in this study. Firstly we performed an 8-person focus group to determine the benefits, challenges, information items and possible actions asked for in the research goals. Following this we performed a questionnaire among 47 participants to determine the relative importance of these benefits, challenges, information items and possible actions. Next, we will explain for both the focus group and the questionnaire, why we chose to use that method of data collection, give details about its execution and discuss how we analyzed the data we gathered.

#### Focus Group

Kontio et al. [Kon04] describe focus groups as *"carefully planned discussions, designed to obtain the perceptions of the group members on a defined area of interest"*. A focus group usually consists of 3 to 12 participants and the discussion is guided and facilitated by a moderator, who follows a predefined structure so that the discussion stays focused. The participants of a focus group are selected via purposive-sampling: they are chosen based on their individual characteristics. The group setting enables the participants to build on the responses and ideas of the others, which increases the richness of the information gained [Lan03]. Strengths of a focus group include the ability to discover new insights, offering the opportunity to explore in-depth why participants think the way they do, and being a cost efficient way of obtaining practitioner experience [Bai78, Kon04]. However, it also shares weaknesses with many other qualitative methods. Firstly, it may be difficult to generalize the results due to the limited number of participants [Jud91, Bai78]. Secondly, group dynamics, communication styles and the social acceptability of certain topics and opinions can influence the discussion and therefore introduce bias [Bai78, Kon04]. Thirdly, it is possible that participants have hidden agendas, for example: trying to come across favorably

[Kon04]. Finally, some of the participants might not completely comprehend the topics discussed [Kon04].

In this study the goal of the focus group was to elicit the benefits, challenges, information items and possible actions. We chose to do a focus group because the method is appropriate for discovering new insights, exploring in-depth why participants think the way they do and because the method requires a limited amount of time of the participants. We performed the focus group in approximately 2.5 hours with 8 people from the site we discussed in the previous subsection. We chose to use participants that actively participated in the use of Communico. We chose these people because they are more likely to have thought about the subject we wished to discuss and because they are motivated to contribute, which is also important in an interactive format such as a focus group. About 2 weeks before the focus group we invited the participants to take part by e-mail invitation. In this e-mail we explained what a focus group entails, what was expected of them and the goal of the focus group. We also emphasized that our interest would lay in their opinions and insights, we would merely be there to observe and moderate and the importance of them all contributing to the discussion roughly equally. Finally, we also sent them a short introduction on what topics we were going to discuss.

During the focus group itself the first two authors were present. One of the authors took the role of moderator, so he made sure the conversations stayed on topic, the structure (see Appendix A) was followed and all participants in the focus group contributed roughly equally to the discussion. The other author mainly took notes and assisted the moderator when necessary. We chose to have the main moderator not take notes because being a moderator requires focus and taking notes can distract him from this activity [Lan03]. In carrying out the focus group we followed a structured approach to ensure we would discuss the topics on which we wanted to elicit opinions. After we introduced the focus group itself and repeated the goals and ground rules, we started by identifying the benefits and challenges of having insight in active conversations. Following this, we identified what information and what actions are important when a conversation is taking place. Finally, we also identified the benefits and challenges of having access to the finished conversations.

The identification of the benefits, challenges, information items and actions was carried out as follows: First we would shortly introduce each subject and subsequently we would hand out sticky notes and ask a question we wanted to know the groups opinion about. Following this, everyone would write answers on sticky notes individually after which we would gather all the sticky notes and discuss each one with the entire group. In the discussion of each sticky note we determined what was meant by it and merged it with, or linked it to various other sticky notes if appropriate, to try and create an overall group consensus.

To conclude the discussion about the focus group we will discuss how we dealt with the challenges of using a focus group to gather data. Firstly, because we defined and followed a predefined structure we were able to control the overall content of the focus group sufficiently and make sure group dynamics did not steer the discussion in an undesirable direction. When a certain discussion did seem to drag on too long without progress the moderator would step in and gently move the discussion onwards. We also dealt with the challenge of social acceptability. For one, we repeatedly emphasized the importance that everyone contributed to the conversation. This point seems to have come across well, as everyone really contributed to the discussions. Another thing we did to deal with this, was the use of sticky notes. Because this method forces everyone to think about a question on their own first and write down their opinions, the temptation to agree with the loudest person or the first person to voice his opinion is reduced. Finally, we performed the focus group in a separate closed office to protect the focus group from outside influences. A third challenge we dealt with concerns that the comprehension of the topic by the participants can be too limited to have an in depth discussion. We dealt with this by choosing motivated participants who had experience with the topic as they used Communico frequently. Next to this, we also sent an introduction into the focus group and the topics discussed in advance and repeated this also in a short presentation right before the focus group started. Finally, the challenge of hidden agendas is not likely to apply in this case study because of the nature of the project. The participants had no logical interest in influencing the outcome of our research as the goals of the research were purely academic with no direct business related decisions depending on it. Overall the focus group worked well and the findings will be discussed in the findings section.

### Survey

Fink [Fin03] describes surveys as: *"a system for collecting information from or about people to describe, compare or explain their knowledge, attitudes or behavior"*. When conducting a survey it is possible to collect information directly, by interviewing people, or indirectly by reviewing written, oral and visual records of people's thoughts and actions. The most used method to do this is a questionnaire [Kel03], in which participants are asked a series of questions for example via filling in a written form, responding to an email or answering the questions on a specifically designed web page. Strengths of gathering data by use of a questionnaire include that the method is quick and requires little effort compared to other methods, that the use of standardized answers simplifies the analysis of data and that respondents can complete the questionnaire when it suits them [Gil00]. Weaknesses are mainly concerned with the quality of the data, both with respect to the completeness and the accuracy. Questionnaires typically have low response rates, have difficulties with motivating the respondents to provide accurate answers, are bounded in the amount and complexity of questions they can ask, are bounded to asking questions and assume people have readily available answers to these questions [Gil00].

In this study we elected to use a questionnaire to determine the mutual importance of the qualitative data we elicited in the focus group. We researched questions such as: *"Which advantages of overhearing conversations are most important?"* and *"Is knowing the subject of a conversation more important when participating in a conversation than when you are merely listening to a conversation?"*. We chose to use a questionnaire to do this because this made it possible for us to include the opinions of a larger group of people than if we used another method and because it is possible to research such questions by using a standardized set of questions. In the questionnaire we asked the respondents to rate the various advantages, challenges, information items and actions on a 5-point Likert scale [Lik32] with a no-opinion option. We included a no-opinion option to prevent people with no opinion on a specific question to answer it anyway and 'pollute' the data in this fashion [Bai78].

In this questionnaire the population is the Exact Online department. We chose to send the questionnaire to the 47 people who installed Communico (out of the 61 that were approached). We chose to send the questionnaire to this sample of the population because we felt these people would be most motivated to complete the questionnaire since they were interested enough to install the tool. Sampling in this fashion to try and achieve a high response rate is known as convenience sampling [Kel03]. We do not think we significantly bias the results by sampling like this because half of the people we sent the questionnaire to (23 out of 47) used the tool for less than 20 hours in a 4-month period. Therefore, it is plausible to assume sufficient people with a general negative view on the concepts are recruited for the sample to accurately represent the population.

We sent the questionnaire (see Appendix B) via e-mail, handed out print outs and also made a web-form available to try and make it as convenient as possible for respondents to return the questionnaire. Other methods we used to maximize the return rate were the personalization of the request, the sending of follow-up requests, asking people in person to get their sympathy and convince them they can make a difference, and the already mentioned convenience sampling. In the end, 44 out of the 47 people we approached returned the questionnaire, so the response rate was 94%. To increase the accuracy of the data we tried to avoid common pitfalls in performing questionnaires like: double-barreled questions, ambiguous questions and leading questions [Bai78]. The results of the questionnaire can be found in anonymized form at `http://Aspic.nl/OCS/QuestionnaireData.xls`.

We analyzed the data as follows: In the questionnaire we asked the respondents to rate the various advantages, challenges, information items and actions. We used this data to reflect on the mutual importance of these. So, for example, we compared the importance of the benefits of overhearing conversations. In order to do so, we applied the Fisher's Least Significant Difference (LSD) method [Fis35] on each of the following categories: benefits of overhearing conversations, challenges of overhearing conversations, information about a conversation, actions possible on a conversation, benefits of finished conversations and challenges of finished conversations. This method first applies the non-parametric Friedman test[1] in

[1] `http://faculty.vassar.edu/lowry/ch15a.html`

order to determine whether the items of the data set of a specific category are significantly different. If the result of applying the Friedman test indicates this is the case, we apply the non-parametric Wilcoxon matched-pairs signed-rank test[2] to pairwise compare all items in that category. From the results of this test it can be concluded whether or not it is likely one of the variables is rated as more important. In the next section we will discuss the results of this analysis.

## 5.4 Findings

In this section we will present the findings of the empirical study. To do this in a clear and concise fashion, we will structure this section in four parts, one for each of the research objectives. For each of these research objectives we will first qualitatively describe our findings. We will describe the benefits, challenges, actions and information items respectively. Following this we will discuss the relative importance of these, by presenting the quantitative data we gathered and our analysis of this data. For each of the comparisons we make we will report the result of the Friedman test and if this test passes we will present a table summarizing all Wilcoxon tests we performed. In this table a green *'larger than'*-sign means the Wilcoxon test passed and the item on the left is rated as more important to a statistically significant level. When the test fails we cannot conclude anything regarding the mutual importance and we show a red *'X'*. In these tables the items that are compared are ordered based on an intermediary ranking of these, produced when applying the Friedman method. The complete results of all tests can be found in non-summarized form at `http://Aspic.nl/OCS/QuestionnaireDataAnalysis.pdf`. In this document the Likert scale values from the questionnaire are represented as a value between 1 and 5, 1 meaning '- -', and 5 meaning '++'.

### 5.4.1 Benefits and Challenges of Overhearing Conversations

#### Benefits

The benefits of having insight in the active conversations we found in the focus group are the following:

| | |
|---|---|
| • Having access to the technical knowledge of colleagues | `Technical Knowledge` |
| • Acquiring involvement with colleagues | `Involvement` |
| • Enjoying your work | `Enjoying` |
| • Being able to join a conversation | `Joining` |

[2]`http://faculty.vassar.edu/lowry/ch12a.html`

- Acquiring insight in the communication structure of the team `Communication Structure`

An overview of how these benefits were rated in the questionnaire is shown in table 5.1.

| | N | | Interquartile range | | |
|---|---|---|---|---|---|
| | Opinion | No-Opinion | 25th | 50th | 75th |
| Technical_Knowledge | 44 | 0 | -/+ | + | ++ |
| Involvement | 44 | 0 | -/+ | + | + |
| Enjoying | 43 | 1 | - | + | + |
| Joining | 44 | 0 | -/+ | + | + |
| Communication_Structure | 42 | 2 | - | -/+ | + |

**Table 5.1:** *Descriptive Statistics - Benefits of overhearing conversations*

When applying the Friedman test on this data set it showed the variables are likely to come from a different distribution ($\chi^2(4) = 9.806, P = 0.044$). So, we applied the Wilcoxon test on all pairs of 2 variables in the data set to check if we could conclude anything about their mutual importance. The results of these tests are summarized in table 5.2. From this table we may conclude that *Technical knowledge* (Z=-2.347, P=0.019), *Involvement* (Z=-2.515, P=0.012) and *Joining* (Z=-2.135, P=0.033) are more important than *Communication structure*.

| | Communication_Structure | Enjoying | Joining | Technical_Knowledge | Involvement |
|---|---|---|---|---|---|
| Communication_Structure | | | | | |
| Enjoying | x | | | | |
| Joining | > | x | | | |
| Technical_Knowledge | > | x | x | | |
| Involvement | > | x | x | x | |

**Table 5.2:** *Comparative Analysis - Benefits of overhearing conversations*

## Challenges

The challenges of having insights in the active conversations we found in the focus group are the following:

| | |
|---|---|
| • It can be distracting from the current work activities | `Distracting` |
| • The context of the conversation can be unclear | `Context` |
| • The information is volatile | `Volatile` |
| • A lack of control for the people whose conversations are overheard | `Lack Of Control` |

An overview of how these challenges were rated in the questionnaire is shown in table 5.3.

| | N | | Interquartile range | | |
|---|---|---|---|---|---|
| | Opinion | No-Opinion | 25th | 50th | 75th |
| Distracting | 43 | 1 | -/+ | + | ++ |
| Context | 43 | 1 | -/+ | + | + |
| Volatile | 44 | 0 | - | + | + |
| Lack_Of_Control | 43 | 1 | - | -/+ | + |

**Table 5.3:** *Descriptive Statistics - Challenges of overhearing conversations*

When applying the Friedman test on this data set it showed the variables are likely to come from a different distribution ($\chi^2(4) = 13.511, P = 0.004$). So, we applied the Wilcoxon test on all pairs of 2 variables in the data set to check if we could conclude anything about their mutual importance. The results of these tests are summarized in table 5.4. From this table we may conclude that *Distracting* and *Context* are more important than *Volatile* and *Lack Of Control*.

| | Volatile | Lack_Of_Control | Context | Distracting |
|---|---|---|---|---|
| Volatile | | | | |
| Lack_Of_Control | x | | | |
| Context | > | > | | |
| Distracting | > | > | x | |

**Table 5.4:** *Comparative Analysis - Challenges of overhearing conversations*

## 5.4.2 Information about a Conversation

The important types of information about a conversation we found in the focus group are the following:

| | |
|---|---|
| • Who are Participating in the conversation | `Participating` |
| • Who are viewing the conversation | `Viewers` |
| • The complete factual content | `Content` |
| • The Commitment of a participant | `Commitment` |
| • The Contribution of a participant | `Contribution` |
| • The subject of the conversation | `Subject` |
| • The tone of the conversation | `Tone` |
| • The type of the conversation | `Type` |
| • The phase the conversation is in | `Phase` |
| • The location the conversation takes place | `Location` |
| • The accessibility of the conversation | `Accessibility` |

In the questionnaire we asked to rate the importance of these information items based on ones involvement in the conversation. Involvement in a conversation has to do with how aware someone is of a conversation and whether he participates in the conversation. Dullemond et. al [Dul10] define a model of conversation involvement based on this, which is shown in figure 5.1.

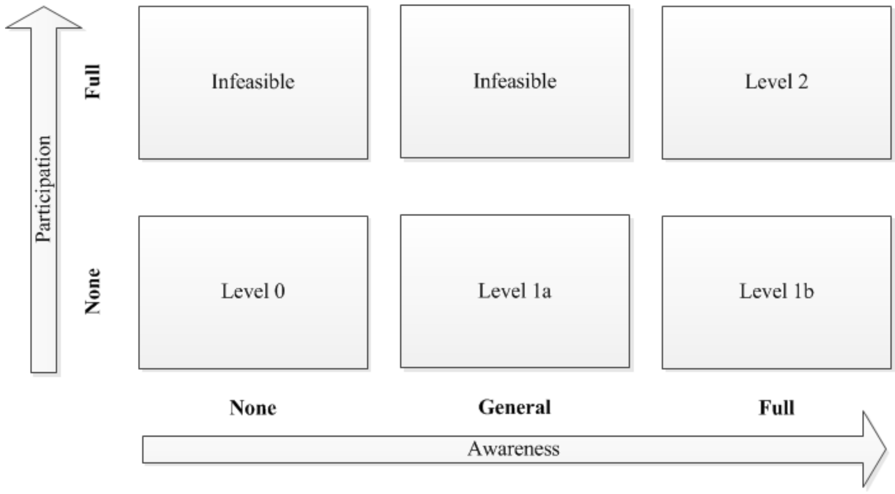


**Figure 5.1:** *Model of conversation involvement [Dul10]*

In the questionnaire we have asked the participants to rate the importance of the various information items in three of these levels:

- Overhearing a conversation `Level 1a`
- Actively listening in on a conversation `Level 1b`
- Participating in the conversation `Level 2`

An overview of this rating is shown in table 5.5.

| | Overhearing | | | | | Listening | | | | | Participating | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | | Interquartile range | | | N | | Interquartile range | | | N | | Interquartile range | | |
| | Opinion | No-Opinion | 25th | 50th | 75th | Opinion | No-Opinion | 25th | 50th | 75th | Opinion | No-Opinion | 25th | 50th | 75th |
| Subject | 43 | 1 | -/+ | + | ++ | 43 | 1 | + | + | ++ | 44 | 0 | + | ++ | ++ |
| Participants | 43 | 1 | -/+ | + | + | 43 | 1 | -/+ | + | ++ | 44 | 0 | + | + | ++ |
| Content | 41 | 3 | -/+ | + | + | 42 | 2 | + | + | ++ | 44 | 0 | + | + | ++ |
| Contribution | 41 | 3 | - | -/+ | + | 42 | 2 | -/+ | + | + | 44 | 0 | + | + | ++ |
| Commitment | 41 | 3 | - | -/+ | + | 41 | 3 | -/+ | + | + | 43 | 1 | + | + | ++ |
| Tone | 41 | 3 | -/+ | + | ++ | 42 | 2 | -/+ | + | + | 43 | 1 | + | + | ++ |
| Type | 41 | 3 | -/+ | + | ++ | 42 | 2 | + | + | ++ | 43 | 1 | + | + | + |
| Accessibility | 39 | 5 | -/+ | + | ++ | 40 | 4 | -/+ | + | ++ | 41 | 3 | -/+ | + | ++ |
| Phase | 41 | 3 | -/+ | -/+ | + | 41 | 3 | -/+ | + | + | 41 | 3 | -/+ | + | ++ |
| Viewers | 41 | 3 | - | - | -/+ | 41 | 3 | - | -/+ | -/+ | 42 | 2 | -/+ | -/+ | + |
| Location | 42 | 2 | - - | - | -/+ | 41 | 3 | -/+ | + | + | 43 | 1 | - - | -/+ | -/+ |

**Table 5.5:** *Descriptive Statistics - Information items*

Because in this case we have two dimensions (the information items and the levels of involvement) instead of one we have analyzed this data more extensively. Firstly, similarly to the analysis in the previous subsection, we have compared the relative importance of the information items in each of the 3 levels of involvement. For all three of these, the Friedman test passed (respectively $\chi^2(10) = 117.712, P = 0.000$, $\chi^2(10) = 122.639, P = 0.000$ and $\chi^2(10) = 101.088, P = 0.000$) and the results of the Wilcoxon comparisons are shown in tables 5.6, 5.7 and 5.8 respectively. From these tables the relative importance of the different information items can be seen for each of the levels of involvement. So, for example, from table 5.6 it can be concluded that when overhearing a conversation the *Participants* are more important than the *Viewers*, *Location*, *Commitment*, *Phase*, *Contribution* and *Accessibility*. Another example is that when participating *Commitment* is more important than *Location*, *Viewers* and *Phase*.

Following this, we also compared the rating of each of these items in the different levels of involvement. The Friedman test failed for the information items *Subject*, *Type*, *Accessibility* and *Phase* (P>0.05), so for this comparison we cannot conclude anything for these items. For the other items we have applied the Wilcoxon test and we found that we could never conclude that an information item is more important in a lower level of involvement when compared to a higher level of involvement. We could however conclude the opposite on multiple occasions. So, a general trend seems to be that information about the conversation is more important when more involved in the conversation. We have presented the results of the Wilcoxon tests in table 5.9.

| | Viewers | Location | Commitment | Phase | Contribution | Accessibility | Content | Tone | Participants | Type | Subject |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Viewers | | | | | | | | | | | |
| Location | x | | | | | | | | | | |
| Commitment | > | > | | | | | | | | | |
| Phase | > | > | x | | | | | | | | |
| Contribution | > | > | x | x | | | | | | | |
| Accessibility | > | > | x | x | x | | | | | | |
| Content | > | > | > | x | > | x | | | | | |
| Tone | > | > | > | x | > | x | x | | | | |
| Participants | > | > | > | > | > | x | x | x | | | |
| Type | > | > | > | > | > | x | x | x | x | | |
| Subject | > | > | > | > | > | > | x | > | x | x | |

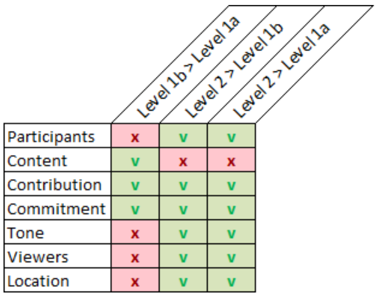
**Table 5.6:** *Comparative Analysis - Information Items of Overhearing a conversation*

| | Location | Viewers | Phase | Accessibility | Commitment | Contribution | Tone | Participants | Content | Type | Subject |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Location | | | | | | | | | | | |
| Viewers | x | | | | | | | | | | |
| Phase | > | > | | | | | | | | | |
| Accessibility | > | > | x | | | | | | | | |
| Commitment | > | > | x | x | | | | | | | |
| Contribution | > | > | x | x | x | | | | | | |
| Tone | > | > | x | x | x | x | | | | | |
| Participants | > | > | > | x | x | x | x | | | | |
| Content | > | > | > | x | x | x | x | x | | | |
| Type | > | > | > | > | > | > | > | x | x | | |
| Subject | > | > | > | > | > | > | > | > | x | x | |

**Table 5.7:** *Comparative Analysis - Information Items of Listening to a conversation*

| | Location | Viewers | Phase | Accessibility | Type | Tone | Commitment | Contribution | Subject | Participants | Content |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Location | | | | | | | | | | | |
| Viewers | > | | | | | | | | | | |
| Phase | > | > | | | | | | | | | |
| Accessibility | > | > | x | | | | | | | | |
| Type | > | > | x | x | | | | | | | |
| Tone | > | > | > | x | x | | | | | | |
| Commitment | > | > | > | x | x | x | | | | | |
| Contribution | > | > | > | x | x | x | x | | | | |
| Subject | > | > | > | x | > | x | x | x | | | |
| Participants | > | > | > | x | > | x | x | x | x | | |
| Content | > | > | > | x | x | x | x | x | x | x | |

**Table 5.8:** *Comparative Analysis - Information Items of Participating in a conversation*



| | Level 1b > Level 1a | Level 2 > Level 1b | Level 2 > Level 1a |
|---|---|---|---|
| Participants | x | v | v |
| Content | v | x | x |
| Contribution | v | v | v |
| Commitment | v | v | v |
| Tone | x | v | v |
| Viewers | x | v | v |
| Location | x | v | v |

**Table 5.9:** *Comparative Analysis - Information Items*

An example of a conclusion we can draw from this table is *Commitment* is more important when listening to a conversation (*Level 1b*) than when overhearing a conversation (*Level 1a*) and more important when participating in a conversation (*Level 2*) than both when listening to a conversation and when overhearing a conversation. So, in general we can see that knowing about the *Commitment* is increasingly more important as the level of Involvement increases.

### 5.4.3 Actions Possible on a Conversation

The actions that are possible with respect to a conversation identified in the focus group are the following:

| | |
|---|---|
| • Joining a conversation | `Joining` |
| • Inviting someone to join a conversation | `Inviting` |
| • Listening to a conversation | `Listening` |
| • Dismissing other participants | `Dismissing Participants` |
| • Dismissing viewers | `Dismissing Viewers` |
| • Acquiring the attention of the participants | `Acquiring Attention` |
| • Notifying others of the conversation | `Notifying Others` |

An overview of how these actions were rated in the questionnaire is shown in table 5.10.

| | N | | Interquartile range | | |
|---|---|---|---|---|---|
| | Opinion | No-Opinion | 25th | 50th | 75th |
| Joining | 44 | 0 | -/+ | + | ++ |
| Inviting | 44 | 0 | + | + | ++ |
| Listening | 44 | 0 | -/+ | + | + |
| Dismissing_Participants | 43 | 1 | - | -/+ | + |
| Dismissing_Viewers | 42 | 2 | - | -/+ | + |
| Acquiring_Attention | 43 | 1 | -/+ | + | ++ |
| Notifying_Others | 42 | 2 | - | + | + |

**Table 5.10:** *Descriptive Statistics - Actions*

When applying the Friedman test on this data set it showed the variables are likely to come from a different distribution ($\chi^2(6) = 51.498, P = 0.000$). So, we applied the Wilcoxon test on all pairs of 2 variables in the data set to check if we could conclude anything about their mutual importance. The results of these tests are summarized in table 5.11.

| | Dismissing_Participants | Dismissing_Viewers | Notifying_Others | Listening | Acquiring_Attention | Joining | Inviting |
|---|---|---|---|---|---|---|---|
| Dismissing_Participants | | | | | | | |
| Dismissing_Viewers | > | | | | | | |
| Notifying_Others | x | x | | | | | |
| Listening | > | x | x | | | | |
| Acquiring_Attention | > | x | x | x | | | |
| Joining | > | > | > | > | x | | |
| Inviting | > | > | > | > | > | > | |

**Table 5.11:** *Comparative Analysis - Actions*

From this table we may conclude that *Inviting* is the most important action. We can also conclude that *Joining* is more important than all the other actions except *Acquiring Attention*. Finally, it is noteworthy that *Dismissing Viewers* is considered more important than *Dismissing Participants*.

### 5.4.4 Benefits and Challenges of Finished Conversations

#### Benefits

The benefits of having insight in the finished conversations we found in the focus group are the following:

| | |
|---|---|
| • Having access to knowledge you might otherwise forget | `Own Knowledge` |
| • Access to technical knowledge of colleagues | `Technical Knowledge` |
| • Acquiring involvement with your colleagues | `Involvement` |
| • Enjoying your work | `Enjoying` |
| • Acquiring insight in the communication structure | `Communication Structure` |

An overview of how these benefits were rated in the questionnaire is shown in table 5.12.

| | N | | Interquartile range | | |
|---|---|---|---|---|---|
| | Opinion | No-Opinion | 25th | 50th | 75th |
| Own_Knowledge | 44 | 0 | -/+ | + | ++ |
| Technical_Knowledge | 44 | 0 | -/+ | + | ++ |
| Involvement | 41 | 3 | - | -/+ | + |
| Enjoying | 43 | 1 | - | -/+ | + |
| Communication_Structure | 43 | 1 | - | -/+ | -/+ |

**Table 5.12:** *Descriptive Statistics - Benefits of finished conversations*

When applying the Friedman test on this data set it showed the variables are likely to come from a different distribution ($\chi^2(4) = 57.331, P = 0.000$). So, we applied the Wilcoxon test on all pairs of 2 variables in the data set to check if we could conclude anything about their mutual importance. The results of these tests are summarized in table 5.13.

| | Communication_Structure | Enjoying | Involvement | Technical_Knowledge | Own_Knowledge |
|---|---|---|---|---|---|
| Communication_Structure | | | | | |
| Enjoying | x | | | | |
| Involvement | > | > | | | |
| Technical_Knowledge | > | > | > | | |
| Own_Knowledge | > | > | > | x | |

**Table 5.13:** *Comparative Analysis - Benefits of finished conversations*

From this table we may conclude that *Own Knowledge* and *Technical Knowledge* are more important than *Communication Structure*, *Enjoying* and *Involvement*.

### Challenges

The challenges of having insight in the finished conversations we found in the focus group are the following:

| | |
|---|---|
| • It can be distracting from the current work activities | `Distracting` |
| • The context of the conversation can be unclear | `Context` |
| • A lack of control for the people whose conversations are overheard | `Lack Of Control` |

An overview of how these challenges were rated in the questionnaire is shown in table 5.14.

| | N | | Interquartile range | | |
|---|---|---|---|---|---|
| | Opinion | No-Opinion | 25th | 50th | 75th |
| Distracting | 44 | 0 | - - | - | + |
| Context | 44 | 0 | -/+ | + | ++ |
| Lack_Of_Control | 43 | 1 | -/+ | + | ++ |

**Table 5.14:** *Descriptive Statistics - Challenges of finished conversations*

When applying the Friedman test on this data set it showed the variables are likely to come from a different distribution ($\chi^2(2) = 31.533, P = 0.000$). So, we applied the Wilcoxon test on all pairs of 2 variables in the data set to check if we could conclude anything about their mutual importance. The results of these tests are summarized in table 5.15.

| | Distracting | Lack_Of_Control | Context |
|---|---|---|---|
| Distracting | ■ | ■ | ■ |
| Lack_Of_Control | > | ■ | ■ |
| Context | > | x | ■ |

**Table 5.15:** *Comparative Analysis - Challenges of finished conversations*

From this table we may conclude that both *Context* and *Lack Of Control* are rated as more important challenges than *Distracting*.

## 5.5 Discussion

In the previous section we have presented the findings of the study we performed. In this section we will first reflect on the results and subsequently discuss limitations of the methods of data collection we used.

### Benefits and challenges of overhearing conversations

In the previous section we have reported on the benefits and challenges of overhearing conversations. In these findings we can see that in general both the benefits and the challenges are rated as important (all but one have a median of '+' in both cases). Therefore further research into the support of overhearing conversations for GSE teams is warranted, however the challenges should be carefully considered. In this consideration, the two challenges rated as most important, *Distracting* and *Lack Of Context*, should be addressed with particular rigor.

### Relevant information about conversations

We have also reported on the information items that are important with respect to conversations. In the findings we see that the relative importance of information items is different in the three different levels of involvement. We can also see that the *Subject* of the conversation is the only information item to be among the most important information items in each level of involvement. Similarly we can see that *Location* is among the lowest rated items in each level of involvement. Finally, it is also noteworthy that, in general, information items seem to gain importance as the involvement of an actor in the conversation increases.

### Possible actions with respect to conversations

We have shown the findings regarding the actions that are important to be able to do with respect to conversations. In the analysis of these we see that *Joining* conversations and *Inviting* others into a conversation are rated as most important. So, with respect to the actions it seems that adding people to a conversation is more important than dismissing them.

### Benefits and Challenges of finished conversations

We have also reported on the benefits and challenges of having access to finished conversations. The main benefits identified here are having access to one's own knowledge and having access to the technical knowledge discussed in the conversations of others. Therefore, we feel it is important to find out how to extract data from conversations to make these easily searchable and help exploit these two benefits. With respect to the challenges it is noteworthy that the interquartile range of the rating of *Distracting* is quite wide. Some of the participants in the questionnaire seem to find being distracted by having access to finished conversations to be a relatively large problem while others find it relatively unimportant. This difference could very well be related to the level of discipline these people possess. In general, however, we can conclude that both *Lack Of Control* and *Context* are rated as more important challenges than *Distracting*.

### Limitations

We provided a quite thorough discussion regarding the limitations of the methods of data collection we used in the description of the research site and method. In this description we also discussed what we did to deal with these challenges. However, there are still some issues that did not fit in that section or need to be emphasized.

Firstly, even though most participants are used to working from home and we had a few participants from Belgium and the USA, most people collaborate mainly in a co-located fashion on a daily basis. Therefore it is possible that the limited exposure to working distributed from their colleagues caused items to be misrated in the questionnaire. For example, as a benefit of overhearing conversations, having insight in the communication structure of the team is rated relatively low. This could of course indicate this is actually the case, however it could also be caused by the fact that participants already knew their colleagues very well, causing having insight into the communication structure to be less important.

A second limitation has to do with the actual size of the sample. For practical reasons, we performed one focus group and sent the questionnaire to 47 people. In this case, as in all cases, increasing the sample size would increase the reliability of the data. For example if the sample size of the questionnaire is increased more Friedman and Wilcoxon tests would provide significant results, allowing for more conclusions to be drawn.

A third limitation has to do with the sample itself. For the focus group we selected all participants from the Delft location, again for practical reasons. Next to this, the people that participated in the focus group also participated in the questionnaire. Both these decisions may have created a bias in the results.

A fourth limitation is that all participants worked for a single department of a single company. When doing research in an attempt to draw conclusions applicable for the general field of software engineering, the sample should resemble that population as accurately as possible. In general we can state that with a larger sample and a more accurate resemblance of the population more externally valid conclusions can be drawn.

Finally, in our analysis of the questionnaire data we used Fishers' Least Significant Difference method to help reduce the number of false positives caused by the pair-wise comparison of all items. In comparison with other methods which aim to accomplish this Fishers' LSD is fairly liberal. We chose to use a fairly liberal method because of the explanatory character of this research. Examples of more conservative methods are: Tukey's Honestly Significant Difference, Scheffe's test and the Bonferroni adjustment.

## 5.6 Concluding Remarks

### 5.6.1 Conclusions

In this empirical study we investigated whether researching how to enable overhearing conversations in a distributed setting is worth pursuing. To acquire the empirical data we performed both a focus group and a questionnaire. In the focus group we identified: (i) the benefits and challenges of having insight in active conversations, (ii) the important types of information about a conversation, (iii) the actions possible on a conversation, and (iv) the benefits and challenges of having access to the finished conversations. Following this we conducted a questionnaire among 47 software engineers to determine the relative importance of these benefits, challenges, information items and possible actions. From the findings of this study we can conclude that overhearing conversations of colleagues is valuable in software engineering teams. Therefore, it is valuable to pursue research about support for overhearing conversations in global software engineering. The main findings of this study are the following:

- All identified benefits and challenges are important
- Distracting and Lack of Context are the most important challenges of overhearing conversations
- Knowing the subject of a conversation is very important
- Knowing the location of a conversation is generally unimportant
- Information about a conversation in general gains importance as the actor is more involved in that conversation
- Adding participants to a conversation is more important than removing them
- It is important to be able to search through past conversations
- Lack of Control and Lack of Context are the most important challenges of having access to finished conversations

Following from these findings, the main contributions of this chapter are:

- The conclusion that research about support for conversations in GSE is worth pursuing
- Insights on important aspects to consider when researching support for conversations in GSE

Directly following from these contributions future work will concern researching how to support the overhearing of conversations in a distributed setting. To do this we will measure how the overhearing of conversations is experienced in such a setting by enabling this with technological support. We will use the data, findings and insights reported in this chapter as a starting point for this.

### 5.6.2 Virtual Office Implications

In this chapter we have reported on an industrial evaluation of being able to overhear conversations of your colleagues in software engineering teams. This study contributes to answering the second research question of this dissertation:

**Research Question 2**

> *What is the value of overhearing conversations in global software engineering?*

In this empirical study we did not identify new requirements of a virtual office. However, the results of this study can be used to determine how the requirements related to communicating in a virtual office can be implemented. Take for example the important types of information about a conversation we identified in this study. The relative importance of these information items can be used to determine what information to display when you are monitoring a conversation (Req 8.).

In this chapter we concluded that overhearing conversations of colleagues is valuable in the field of software engineering. In the next chapter we attempt to answer the last part of research question two. We present a technological implementation which enables the overhearing of conversations in a distributed setting, and we measure the value of overhearing conversations in global software engineering.

Chapter 6

# An Industrial Evaluation of an Open Conversation Space

*Software engineering is by nature a highly collaborative activity. However, collaborating effectively in global software engineering, in which team members are geographically, temporally and/or socio-culturally separated from each other, is more difficult. In a traditional co-located setting, one of the most important communication patterns is a (face-to-face) conversation. Technological solutions to have conversations in a distributed setting are commonly used, however overhearing conversations of others is not explicitly supported. In this chapter we report on the evaluation of supporting overhearing conversations with technology in a distributed industrial setting. To do this we first present a tool we developed which fulfills the requirements of an Open Conversation Space: Communico. Next, we deployed this tool in an international software development company. Based on this evaluation we report lessons learned and conclude with the most important findings of this study.*

This chapter is based on both our publication *"Overhearing Conversations in Global Software Engineering - Requirements and an Implementation"* in the Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2011), and on our publication *"An Industrial Evaluation of Technological Support for Overhearing Conversations in Global Software Engineering"* in the Proceedings of the 7th International Conference on Global Software Engineering (ICGSE 2012). The first publication is co-authored by both Kevin Dullemond and Rini van Solingen. The second publication is only co-authored by Kevin Dullemond.

## 6.1 Introduction

There are strong indications that the ability to overhear conversations is valuable in carrying out your work as a software engineer, see chapter 5. In distributed settings however, overhearing conversations is infeasible without technological support and thus far there is no empirical evidence that it can be supported by technology in a satisfying way. In fact, there are no documented case studies researching the overhearing of conversations in such a setting at all. Therefore, to gather such empirical data, we performed a case study during a period of four months at an international software development company in which Communico was used. Communico is a tool we developed which fulfills the requirements of an open conversation space. The goal of this study is:

*"To measure the value of overhearing conversations in global software engineering from actual industrial experience"*

This chapter is structured as follows. In the next section we discuss the background of this research. In section 6.3 we discuss existing tools for communicating in a distributed setting. Subsequently, in section 6.4, we discuss Communico, a virtual open conversation space we developed. In section 6.5 we discuss the research site and methods of data collection and analysis. Subsequently we present our findings in section 6.6 and reflect upon these results in section 6.7. Next we discuss threats to validity in section 6.8. Finally, we conclude upon our work and discuss future research in section 6.9.

## 6.2 Background

More and more collaborative software engineering is no longer conducted in a single office building but in multiple dislocated office buildings or even from home. This is caused by the increasing globalization of business [Car99, Her01, Her07] and the rising popularity of working from home [Die09]. In collaborative work having access to the knowledge about the context in which you are working (commonly referred to as *'awareness'* [Sch02, Dou92]) is essential to properly collaborate with others [Sch02, Syr97]. However, when collaborating physically separated from each other team members can no longer exchange information without technological support because they do not share a common work environment. So, technological support is required to be able to gather and maintain sufficient awareness to be able to collaborate. However, the information exchange realized with technological support (e.g. telephone or e-mail) is generally inferior to information exchange in a traditional co-located setting because in comparison it (i) takes more effort since the communication is more intentional [Gut04], (ii) is more obtrusive [Fog05], (iii) happens less frequently [Her03, All77] and (iv) contains less information [Gut04, Ols96]. Therefore it is important to research and develop technological support for sharing awareness in a fashion that is as unobtrusive and effortless as possible while still providing rich and recent information.

Conversations are one of the most important communication patterns that occur in a traditional office setting [Per94]. In chapter 4 we defined a conversation in the context of global software engineering as: *"An exchange of information between two or more people where those participating use synchronous communication directed at the other participants"*. Next to this definition we also discussed the importance of conversations and the importance of being able to overhear conversations of others. Finally, we introduced the concept of an open conversation space to refer to such a setting and defined a set of requirements such a space should implement.

## 6.3 Related Work

Before we discuss Communico in the next section, we will first discuss existing tools for communicating in a distributed setting and compare these with Communico based on whether and how they implement the set of requirements of an open conversation space. Firstly, Communico is a virtual open conversation space, so it is different from tools which are not suitable to have conversations and tools with which it is not possible to overhear conversations of others. As explained in the previous section, an information exchange can only be considered a conversation when it is both synchronous and directed at two or more participants. Therefore tooling not suitable for synchronous communication (e.g. e-mail and forums) or directed communication (e.g. forums and Twitter [Jav07]) does not support having conversations and therefore fulfills none of the five requirements. Other tools which do support having conversations do not support the detection and monitoring of these conversations by others (e.g. Instant Messaging, telephone and Google Wave [Tra08]) and therefore are not virtual open conversation spaces as well since they do not implement **REQ2** and **REQ3**.

There are also existing solutions which fulfill all of the five requirements. We have performed a comparison between a number of these by comparing how the different tools implement the requirements. An overview of the comparison is shown in table 6.1 (**REQ4** and **REQ5** are omitted in this table because we could not define an orthogonal subdivision in which these requirements are divided). In this table it can be seen that the existing solutions mainly differ in three ways. Firstly, most of the tools on the list support indirect conversation initiation: they support starting a conversation by creating a topic. In three of these tools however, conversations can be initiated directly by starting to talk to a specific person. Secondly, while all the virtual open conversation spaces we looked at support manual detection of ongoing conversations, only Reachout also supports the automatic detection of the conversations by subscribing to certain topics. Finally, with the exception of Internet Relay Chat and VirtualOffice, all solutions make the conversations explicit. In IRC and VirtualOffice however, all messages are shown in sequential order irrespective of what conversation they belong to.

| | **REQ1:** Initiating | | **REQ2:** Detecting | | **REQ3:** Monitoring | |
|---|---|---|---|---|---|---|
| | *Direct* | *Indirect* | *Manual* | *Automatic* | *Explicit* | *Implicit* |
| Internet Relay Chat (IRC) | ✓ | ✓ | ✓ | | | ✓ |
| VirtualOffice [Sha11] | ✓ | ✓ | ✓ | | | ✓ |
| GroupBanter [Ink08] | | ✓ | ✓ | | ✓ | |
| Babble [Eri99] | | ✓ | ✓ | | ✓ | |
| Loops [Eri06] | | ✓ | ✓ | | ✓ | |
| ReachOut [Rib02] | | ✓ | ✓ | ✓ | ✓ | |
| Threaded Chat [Smi00] | | ✓ | ✓ | | ✓ | |
| OpenMessenger [Bir08] | ✓ | | ✓ | | ✓ | |
| Communico | ✓ | | ✓ | ✓ | ✓ | |

**Table 6.1:** *Comparison of Virtual Open Conversation Spaces*

Based on this analysis we have implemented Communico making use of the strengths of the solutions we looked at. In the design of Communico we have attempted to mimic the traditional office setting as much as possible because it is clear that co-located software engineering has awareness benefits and therefore it seems like a good starting point to mimic this as much as possible [Omo09]. So in comparison with the solutions compared in table 6.1 we have made the following design decisions: Firstly, we have chosen to use direct conversation initiation by starting to talk with one or more specific people because in our opinion this is the most common way to initiate a conversation in a traditional office setting. Secondly, we have chosen to support both the manual and the automatic detection of conversations because both occur in the traditional office setting as well: people both overhear conversations by actively looking around and by being triggered by a certain event while carrying out another task [Mar02]. Finally we have chosen to make conversations explicit. We did this because using implicit conversations, like in IRC, limits the creation of a structured and logical layout for group discussions [Tra08]. In a co-located setting, conversations are also sequential in nature, but other indicators help to more easily identify conversations as such (e.g. the placement of individuals in the room and the people at whom people are looking). When only the sequential ordering of messages is known, all that is left to identify conversations is semantics. Because of this it requires more effort to be aware what conversations are going on.

We chose to develop and subsequently evaluate Communico because it is conceptually different from the existing technological solutions. We will provide a detailed description of Communico in the next section.

## 6.4 Communico

In this section we will discuss the improved version of Communico; a virtual open conversation space we developed (See [Dul11a] for a video demonstration). To do this, we will first briefly look at the technical implementation of Communico. Following this we will discuss how Communico implements the requirements of an open conversation space which we discussed in chapter 4.

### 6.4.1 Technical Implementation

Communico uses Microsoft Office Communications Server[1] to support having conversations in a distributed setting. The Office Communications platform supports text-based communication (Instant Messaging), audio conferencing, video conferencing and screen sharing. We chose to use this platform because it supports all these features and because its popularity in industry makes it easier to find a setting for performing a case study. Because we build upon an existing communication solution rather than develop our own, Communico can gradually be introduced into a business. This is because people using Communico and people using Office Communicator can directly communicate with each other. This does imply that only conversations in which at least one participant is using Communico are visible in the open conversation space and only people using Communico have access to this space. Even so, allowing for gradual introduction of a communication tool into a business environment eases its introduction.

In Communico we chose to initially only incorporate the IM-conversations in the open conversation space. We did this for three reasons. Firstly, an Instant Messaging tool supports having conversations because it allows for synchronous communication[2] which is directed at the other participants in the conversation. Secondly, textual information exchange is, in comparison with audio and video, much easier to record, analyze and search through in an automated fashion. Finally, text based communication is commonly used to achieve awareness and is very simple to use [Gut04]. So, focusing on the IM-conversations seems like a good starting point in creating a virtual open conversation space.

The technical infrastructure of Communico is shown in figure 6.1. When using Office Communications Server as a communication platform, a central server routes all communication and every user of the system runs a client (in this case Office Communicator 2007 R2) to use the functionality this server offers. A Communico client runs alongside Office Communicator on the machine of everyone using Communico and all these instances are connected to a central Communico server. The Communico clients use the Office Communicator Automation API to gather data about users and conversations and send this information to the central Communico server. This server maintains a complete model of all data and sends updates to all clients when this model changes. These updates can cause the Communico clients to update its user interface or initiate a certain action, via the Office Communicator Automation API, like inviting someone into a conversation.

[1] `http://www.microsoft.com/communicationsserver/en/in/`

[2] We regard this as synchronous because the sending and receipt of messages can be regarded as instantaneous.

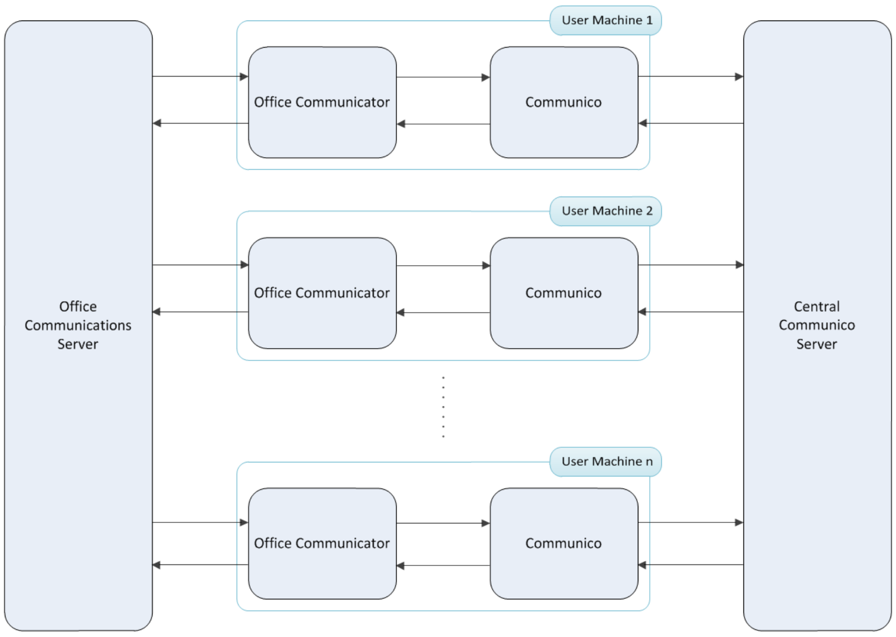


**Figure 6.1:** *Technical Infrastructure*

### 6.4.2 A Virtual Open Conversation Space

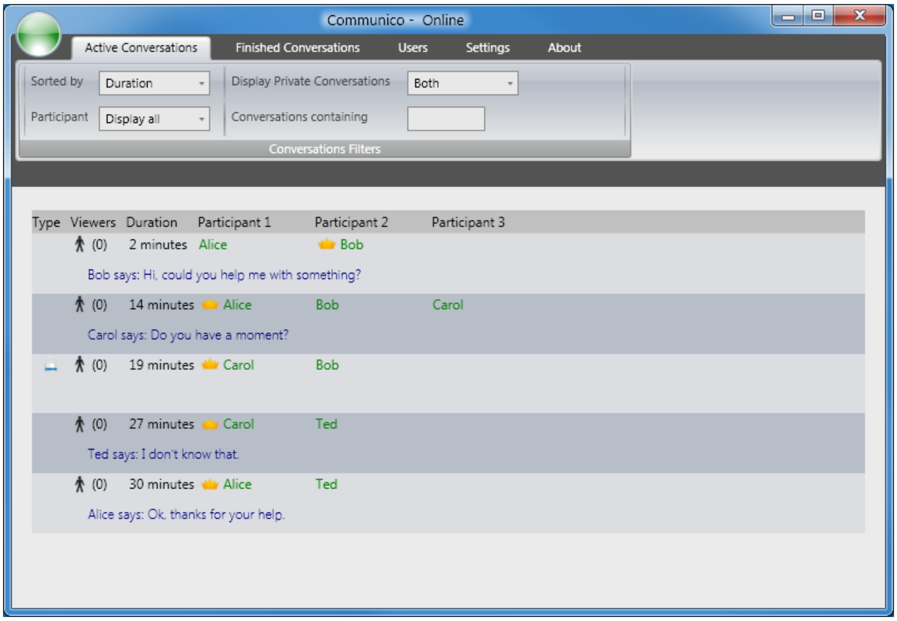
In this section, we will discuss Communico by going over the requirements of an open conversation space.

#### Uninitialized Conversation

In Communico conversations are initiated by double clicking the name(s) of the user(s) you wish to start a conversation with (**REQ1**). Because this is a direct way to initiate a conversation it mimics walking over to someone and starting to talk to him or her in a co-located setting. We chose this method of conversation initiation for two reasons. Firstly, in our opinion this is the most common way to initiate a conversation in a co-located situation. Secondly, starting a conversation in this fashion is common practice in IM-tools including Office Communicator. The choice for direct conversation initiation, however, does not rule out the introduction of an indirect form of conversation initiation (for instance topic based) in the future.

### Active Conversation

Three requirements of an open conversation space are related to an active conversation. Users should be able to find out about conversations (**REQ2**), listen to conversations of others (**REQ3**) and become part of a conversation (**REQ4**). The two ways of finding out about a conversation are actively looking for it (i.e. by looking around in a traditional office setting) and automatic detection (i.e. subconsciously picking up on an interesting conversation while working). In Communico the former is implemented in the active conversations tab depicted in figure 6.2. On this tab a list is shown of all conversations that are currently going on between the members of the project team. For each conversation basic information is shown like the participants, the last thing said, and the number of viewers. A viewer of a conversation is a user that is looking at the detailed information about that conversation (discussed later). The list of active conversations can also be sorted (e.g. the conversation with the latest message on top) and filtered (e.g. only show conversations containing a certain word or phrase) to help the user discover interesting conversations.

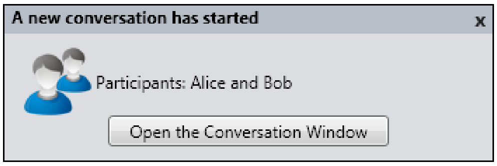


Sorted by: Duration
Participant: Display all
Display Private Conversations: Both
Conversations containing:

| Type | Viewers | Duration | Participant 1 | Participant 2 | Participant 3 |
|---|---|---|---|---|---|
| | (0) | 2 minutes | Alice | Bob | |
| | Bob says: Hi, could you help me with something? | | | | |
| | (0) | 14 minutes | Alice | Bob | Carol |
| | Carol says: Do you have a moment? | | | | |
| | (0) | 19 minutes | Carol | Bob | |
| | (0) | 27 minutes | Carol | Ted | |
| | Ted says: I don't know that. | | | | |
| | (0) | 30 minutes | Alice | Ted | |
| | Alice says: Ok, thanks for your help. | | | | |

**Figure 6.2:** *Active Conversations Tab*

Communico implements the second way of finding out about conversations, the automatic detection of conversations, with the use of desktop alerts (Figure 6.3). Users can configure Communico to display a desktop alert if an active conversation meets a certain criteria to prevent information overload. Supported criteria for showing desktop alerts are: that a conversation (i) contains a certain key word, (ii) has a specific participant, (iii) has a certain number of viewers or (iv) is running for a certain amount of time. A disadvantage of using desktop alerts is that active notification by the system potentially disrupts the users [Iqb07, Cut01, Cze04]. However, we argue this is also the case in the traditional co-located setting, as people are also disrupted by conversations that are in fact not that interesting to them. The goal of the configurable criteria is to emulate the implicit thought process in the traditional office setting. Besides this, the field study reported in Iqbal and Horvitz [Iqb10] also indicates the awareness gained could be worth the added disruptions caused by the desktop alerts.

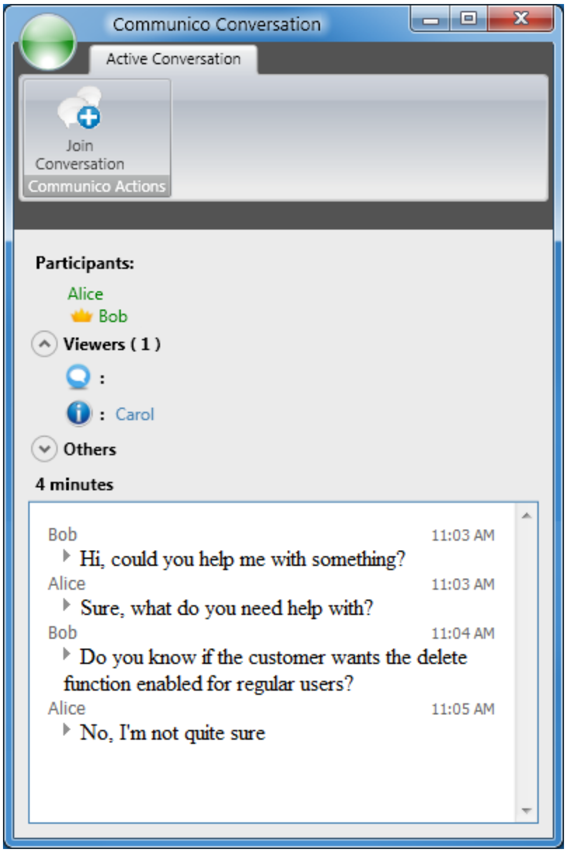


Participants: Alice and Bob

**Figure 6.3:** *Desktop Alert*

After a user finds out about a conversation that is potentially interesting he can, like in a traditional office setting, start to actively listen to the conversation. In Communico this is done either by double clicking the conversation in the active conversation tab or by clicking the desktop alert about the conversation. When this is done a window showing more detailed information about that conversation, like shown in figure 6.4, is opened.

When viewing the detailed information about the conversation a user can also see everything that has been said in the conversation and information regarding the involvement of others in it. We choose to show the involvement of others here because (i) we regard it an integral part of a conversation, (ii) involvement information is also available in the traditional office setting and (iii) a number of sources (e.g. [Smi00, Tra05]) also report on its importance. In the traditional office setting the involvement of someone in a conversation depends on how aware he is of the conversation and whether or not he participates in it. Therefore, we show all people that are currently participating and all people that are currently monitoring the conversation (by accessing the detailed information). Next to this, we also show who monitored and participated in the conversation in the past because we think past involvement is also important when monitoring a conversation. By implementing these levels of involvement, Communico conforms to the model of conversation involvement, see figure 5.1 in the previous chapter.

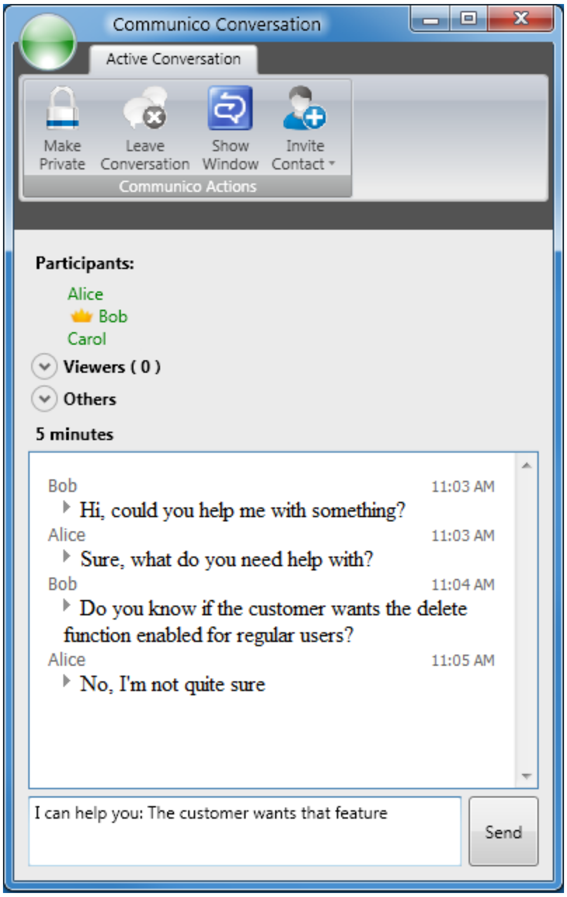


Participants:
Alice
Bob
Viewers ( 1 )
:
: Carol
Others
4 minutes

Bob 11:03 AM
Hi, could you help me with something?
Alice 11:03 AM
Sure, what do you need help with?
Bob 11:04 AM
Do you know if the customer wants the delete function enabled for regular users?
Alice 11:05 AM
No, I'm not quite sure

**Figure 6.4:** *Conversation Window - Viewing*

People can join a conversation either by being invited by people already participating or by viewing the conversation and requesting to join it by clicking the join button. When such a join request is accepted by the owner of the conversation, the user changes from merely viewing the conversation to being a participant, resulting in the conversation window depicted in figure 6.5. As a participant, the user can actively contribute to the conversation. Next to this, all participants also have the option to make a conversation private, effectively hiding the content from all users that are not participating in the conversation. This functionality was added to mimic going to a separate office to talk in private in the traditional office setting.



Participants:
Alice
Bob
Carol
Viewers ( 0 )
Others
5 minutes

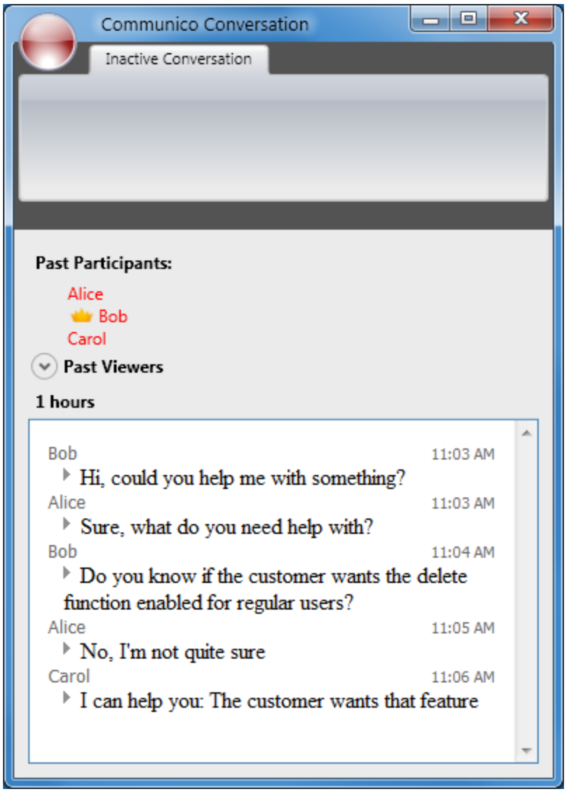
Bob 11:03 AM
Hi, could you help me with something?
Alice 11:03 AM
Sure, what do you need help with?
Bob 11:04 AM
Do you know if the customer wants the delete function enabled for regular users?
Alice 11:05 AM
No, I'm not quite sure

I can help you: The customer wants that feature

**Figure 6.5:** *Conversation Window - Participating*

### Finished Conversation

A conversation is a synchronous exchange of information and therefore its explicit end is when the synchronous communication ceases. This can however be difficult to detect because it can be unclear whether the synchronous communication has ended or all participants are simply thinking about their next reply. Because a conversation requires two or more people to communicate, in Communico we chose to define the end of a conversation as the moment the total number of participants in the conversation becomes one or zero (**REQ5**). So when two people participate in a conversation and one of them leaves it, either by clicking leave conversation or closing the conversation window, the conversation finishes. When a conversation finishes it becomes immutable: people can no longer join and as a result no content can be added to the conversation as well. An example of a finished conversation is shown in figure 6.6.



Past Participants:
Alice
Bob
Carol
Past Viewers
1 hours

Bob 11:03 AM
Hi, could you help me with something?
Alice 11:03 AM
Sure, what do you need help with?
Bob 11:04 AM
Do you know if the customer wants the delete function enabled for regular users?
Alice 11:05 AM
No, I'm not quite sure
Carol 11:06 AM
I can help you: The customer wants that feature

**Figure 6.6:** *Finished Conversation Window*

Finished conversations are not discarded, but instead made available in Communico's finished conversations tab because past conversations are a valuable source of information [Nii08]. In a normal office setting team members each have access to part of this information, namely the conversations which they participated in or overheard. With Communico however, team members have access to all conversations that occurred in the open conversation space, can access them in full detail and can automatically search through them as well.

## 6.5 Research Site and Method

In the introduction we defined the research goal of this study: *To measure the value of overhearing conversations in global software engineering from actual in-*

*dustrial experience.* To reach this goal we use Communico to enable the overhearing of conversations in an industrial case setting. In the previous chapter we identified (i) benefits and challenges of having insight in active conversations, (ii) important types of information about a conversation, (iii) actions that are possible with respect to a conversation, and (iv) benefits and challenges of having insight in finished conversations. In this chapter we use these identified benefits, challenges, information items and actions to investigate the following four research questions:

- How well are the benefits and challenges of having insight in active conversations exploited and alleviated?
- How well are the conversations represented?
- How well are actions to be carried out on a conversation supported?
- How well are the benefits and challenges of having insight in finished conversations exploited and alleviated?

In the remainder of this section we describe the industrial case setting and the methods with which we have investigated the research questions.

### 6.5.1 Site

Participants of this study are the same group of software engineers at Exact as those in the study presented in the previous chapter. Exact is a software development company operating in 40 countries and offers Enterprise Resource Planning software for medium-sized and small businesses. The specific group of employees that are involved in the study consists of people who work on a product called Exact Online. The majority of the people involved in this study (42), work from the Delft office in The Netherlands, but all work from home often. Next to this, also three people participated from the Wemmel office in Belgium and two from the Minneapolis office in the USA. The majority of the people that participated in the study use Instant Messaging software on a daily basis, even when working from the same office. During the case study, which lasted four months, they also used Communico to be able to overhear conversations of the rest of the group.

### 6.5.2 Data Collection and Analysis Methods

To reach our research objectives we used four methods to acquire the empirical data in this study: a focus group, a semi-structured interview, a questionnaire and transactional log analysis.

#### Focus Group

We performed a focus group [Kon04] to gather insights, ideas, viewpoints and opinions of people who frequently used Communico in a practical case setting. One of the main advantages of such a group setting is that it enables the participants to build on the responses and ideas of others, which increases the richness of the information gained [Lan03].

The focus group we performed lasted approximately 2.5 hours and we selected eight frequent users of Communico from the Delft office to participate in it. Selecting participants based on their individual characteristics like this is known as purposive-sampling. We chose these people because they are more likely to have thought about the subject we wished to discuss and because they are motivated to contribute. In carrying out the focus group we followed a structured approach (see Appendix C) to ensure we would discuss the topics on which we wanted to elicit opinions. The focus group itself was conducted in a separate closed office to protect the focus group from outside influences.

#### Semi-Structured Interview

We performed semi-structured interviews [Fon05] to gather insights, ideas, viewpoints and opinions of the interviewees. We performed two semi-structured interviews, one for the people from the Belgium office and one for the people from the US office because they could not attend the focus group. Their input is particularly valuable because the main goal of this research is to investigate the value of overhearing conversations in a distributed setting and these people worked most distributed from their colleagues. In these interviews we used the same structured approach as in the focus group.

#### Questionnaire

We chose to use a questionnaire [Fin03] because this method makes it feasible to include the opinions of a relatively large group of people by using a standardized set of questions. In the questionnaire (see Appendix D) we asked the respondents to rate their experience in the case study on a 5-point Likert scale [Lik32]. We included a *'no-opinion'* option to prevent people with no opinion on a specific question to answer it anyway and 'pollute' the data in this fashion [Bai78].

We applied the Fisher's Least Significant Difference (LSD) method [Fis35] on the ratings in each of the researched categories to reflect on their mutual importance. This method first applies the non-parametric Friedman test[3] in order to determine whether the items of the data set of a specific category are significantly different. If the result of applying the Friedman test indicates this is the case, we apply the non-parametric Wilcoxon matched-pairs signed-rank test[4] to

[3]http://faculty.vassar.edu/lowry/ch15a.html
[4]http://faculty.vassar.edu/lowry/ch12a.html

pairwise compare all items in that category. From the results of this test it can be concluded whether or not it is likely one of the variables is rated as more significant.

We sent the questionnaire to 47 members of the Exact Online department of which 44 returned the survey (94% response rate). In the survey we asked people whether they frequently used Communico and only allowed users that indicated they did fill out the remaining questions. In total this concerned 25 people. The results of the questionnaire can be found in anonymized form at `http://Aspic.nl/VOCS/QuestionnaireData.pdf`.

### Transactional Log Analysis

Transactional log analysis is a data collection method for analyzing system performance and user behavior [Jan08]. The main benefits of using transaction log analysis to analyze user behavior are that it is an unobtrusive method and gathers much more data than any data set obtained via surveys, laboratory studies or by user observation in naturalistic settings [Jan08]. We use the method in this study to gather data on usage frequency of the conversation overhearing functionality to be able to reflect on the adoption rate of our solution.

## 6.6 Findings

We present the findings of the empirical study in five parts. Firstly, we reflect on the four research questions. We do this by reiterating both the benefits and challenges of overhearing both active and finished conversations, and the information items and actions identified in the previous chapter. Subsequently, we check for each of these characteristics whether we can draw statistically valid conclusions regarding the relative ordering of how well they are implemented in Communico. If this is the case we show their relative ordering by presenting a table summarizing all comparisons. In this table a (green) *'larger than'*-sign means the item on the left is rated as more important to a statistically significant level. We show a (red) *'X'* when we cannot draw statistically significant conclusions regarding the relative ordering of two items. Finally, we analyze the user behavior: how the users of Communico interacted with the system. For each of the five parts, we discuss the findings, discuss possible improvements of Communico and present lessons learned. For each of the characteristics, a detailed presentation can be found at `http://Aspic.nl/VOCS/QuestionnaireDataAnalysis.pdf` which includes all of the data we gathered, a descriptive analysis of this data illustrating its distribution and a complete analysis to determine the relative ordering.

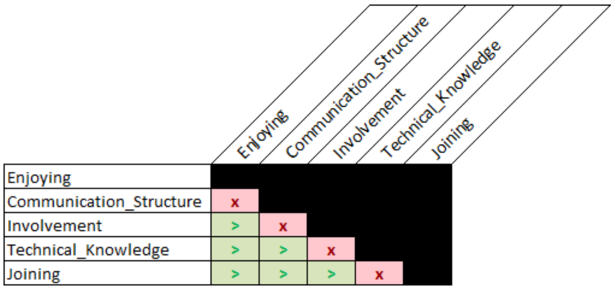
### 6.6.1 Benefits and Challenges of Overhearing Conversations

#### Benefits

The benefits of having insight in the active conversations we identified in chapter 5 are the following:

| | |
|---|---|
| • Having access to the technical knowledge of colleagues | `Technical Knowledge` |
| • Acquiring involvement with colleagues | `Involvement` |
| • Enjoying your work | `Enjoying` |
| • Being able to join a conversation | `Joining` |
| • Acquiring insight in the communication structure of the team | `Communication Structure` |

When applying the Friedman test on the gathered data it showed the variables are likely to come from a different distribution ($\chi^2(4) = 29.651, P = 0.000$). So, we applied the Wilcoxon test on all pairs of 2 variables in the data set to check if we could conclude anything about their mutual importance. The results of these tests are summarized in table 6.2.



| | Enjoying | Communication_Structure | Involvement | Technical_Knowledge | Joining |
|---|---|---|---|---|---|
| Enjoying | | | | | |
| Communication_Structure | x | | | | |
| Involvement | > | x | | | |
| Technical_Knowledge | > | > | x | | |
| Joining | > | > | > | x | |

**Table 6.2:** *Comparative Analysis - How well Communico exploits the benefits of overhearing conversations*

From table 6.2 we may conclude that *Joining* is exploited more than *Involvement* (Z=-2.725, P=0.006), *Communication_Structure* (Z=-3.397, P=0.001) and *Enjoying* (Z=-3.502, P=0.000). Next to this, we can conclude that *Technical_Knowledge* is exploited more than *Communication_Structure* (Z=-2.517, P=0.012) and *Enjoying* (Z=-3.147, P=0.002). Finally, we can also conclude that *Involvement* is exploited more than *Enjoying* (Z=-2.696, P=0.007).

In the focus group the benefits reported to be exploited best by Communico were *Joining* and *Technical_Knowledge*. In the interviews these two were also seen as well exploited by Communico, however the interviewees also reported a significant increase in *Involvement* with the rest of the team due to being able to overhear their conversations with Communico. One of the interviewees from

Belgium said: *"I felt more like being there"* while one of the interviewees from the USA said: *"In the morning I would scroll through all the conversations my colleagues had during their working day that far. This made me feel more connected to them"*. It is likely the interviewees felt different than the focus group participants because the people that were interviewed worked dislocated from nearly all of their colleagues while the distribution was significantly less for the participants of the focus group.

| **Lesson Learned 1** |
|---|
| Overhearing conversations with technological support results in a stronger increase of involvement with colleagues for people that work more dislocated from their colleagues than for those that work more co-located |

Finally, participants of the focus group reported the *Communication_Structure* as not being exploited particularly well. They suggested making the relations explicit in the form of a graph, since manually checking who are communicating often takes too much effort. It is noteworthy that with respect to the benefits only a possible improvement was mentioned in relation to the exploitation of the *Communication Structure* while in chapter 5 this is rated as one of the least important benefits.

| **Lesson Learned 2** |
|---|
| Only showing the conversations is not sufficient to acquire insight in the communication structure of the team |

### Challenges

The challenges of having insight in the active conversations we identified in chapter 5 are the following:

| | |
|---|---|
| • It can be distracting from the current work activities | `Distracting` |
| • The context of the conversation can be unclear | `Context` |
| • The information is volatile | `Volatile` |
| • A lack of control for the people whose conversations are overheard | `Lack Of Control` |

In the focus group and interviews it was discussed that *Volatile* is not a large problem since this is something inherently tackled by a tool such as Communico which saves conversations. The other three challenges of overhearing conversations are however also encountered when using Communico. The focus group identified several improvements of Communico to help alleviate the two challenges rated as most important in the previous chapter; *Distracting* and *Context*. With respect to *Distracting* the participants of the focus group suggested making it possible to withdraw yourself from the open conversation space completely by putting on *'virtual head phones'*. Subsequently, when you take off these *'virtual head phones'* it should be possible to get some sort of summary of the conversations that occurred while you were occupied. Another method proposed was to limit the amount of conversations you overhear by creating virtual office walls separating yourself from certain groups of people while still overhearing the conversations of others.

| **Lesson Learned 3** |
|---|
| Having control over the amount of conversations you overhear is important to limit distractions |

Further, the focus group discussed that the relations between conversations are an important part of the context of a conversation and suggested to make it possible to either automatically or manually link related conversations to each other.

| **Lesson Learned 4** |
|---|
| Making the relations between conversations clearer improves the clarity of the context of a conversation |

Finally, the participants argued there was insufficient control with respect to making conversations private. Firstly, they argued they wanted to be able to make conversations private before they start and secondly that they would like to see private conversations removed from the active conversations list altogether. Currently, private conversations are shown there but their content is hidden from non-participants. Participants argue they see no value in knowing others are having a private conversation and they do not want others to know of their own private conversations as well.

| **Lesson Learned 5** |
|---|
| Only enabling making conversations private after initialization provides insufficient control for the people whose conversations are overheard |

| **Lesson Learned 6** |
|---|
| Making the fact that private conversation are occurring visible to non-participants provides insufficient control for participants in private conversations |

### 6.6.2 Information about a Conversation

The important types of information about a conversation we identified in chapter 5 are the following:

| | |
|---|---|
| • Who are Participating in the conversation | `Participating` |
| • Who are viewing the conversation | `Viewers` |
| • The complete factual content | `Content` |
| • The Commitment of a participant | `Commitment` |
| • The Contribution of a participant | `Contribution` |
| • The subject of the conversation | `Subject` |
| • The tone of the conversation | `Tone` |
| • The type of the conversation | `Type` |
| • The phase the conversation is in | `Phase` |
| • The location the conversation takes place | `Location` |
| • The accessibility of the conversation | `Accessibility` |

The results of these Wilcoxon tests are summarized in table 6.3.

| | Location | Tone | Commitment | Phase | Type | Subject | Contribution | Content | Accessibility | Viewers | Participants |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Location | | | | | | | | | | | |
| Tone | x | | | | | | | | | | |
| Commitment | x | x | | | | | | | | | |
| Phase | > | > | > | | | | | | | | |
| Type | > | > | > | x | | | | | | | |
| Subject | > | > | > | x | x | | | | | | |
| Contribution | > | > | > | x | x | x | | | | | |
| Content | > | > | > | x | x | x | x | | | | |
| Accessibility | > | > | > | > | > | > | > | x | | | |
| Viewers | > | > | > | > | > | > | > | x | x | | |
| Participants | > | > | > | > | > | > | > | > | x | x | |

**Table 6.3:** *Comparative Analysis - How well Communico shows the information items about a conversation*

The focus group and interviews elicited similar opinions about the importance of the information items. Both considered it important to be able to tag conversations with the *Subject* to be able to quickly and unobtrusively decide whether it is interesting. In the previous chapter the *Subject* was also elicited as the most important information item to accomplish this and in the current version of Communico it was rated relatively low (see table 6.3). One of the participants said: *"I used the join feature less than would have been possible if the automatic detection of conversations had been better"*.

| **Lesson Learned 7** |
|---|
| Deciding whether a conversation is interesting can be done quicker and less obtrusive when its subject is known |

Next to this, showing the last two sentences instead of the last one can also improve conversation detection since the last sentence is often an acknowledgment like: *'sure'*, *'alright'* or *'I'll get right on that'*. As a final way to improve conversation detection, some participants proposed to include a text based sliding text ticker to find out about a conversation in Communico. The advantage of such a ticker is that it is less disruptive than a desktop alert while being easier to continually scan than the active conversations list.

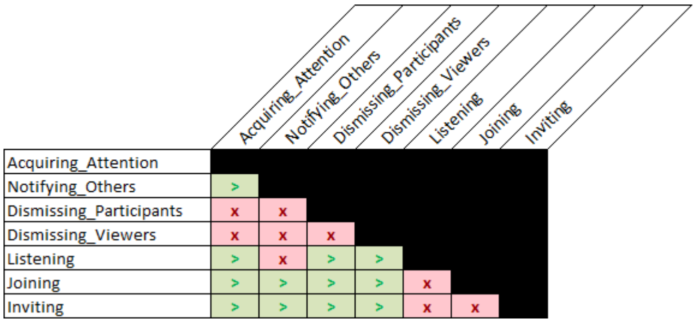
### 6.6.3 Actions Possible on a Conversation

The actions that are possible with respect to a conversation we identified in chapter 5 are the following:

| | |
|---|---|
| • Joining a conversation | *Joining* |
| • Inviting someone to join a conversation | *Inviting* |
| • Listening to a conversation | *Listening* |
| • Dismissing other participants | *Dismissing Participants* |
| • Dismissing viewers | *Dismissing Viewers* |
| • Acquiring the attention of the participants | *Acquiring Attention* |
| • Notifying others of the conversation | *Notifying Others* |

In the focus group it was discussed that while *Joining* is supported adequately (see table 6.4) it would be preferable for the join-process to be more like the co-located setting. Currently, someone has to request to join the conversation and explicit permission needs to be given for this to be allowed. The participants of the focus group suggested changing this so that people automatically join a conversation when they start to talk in a conversation they are watching. One of the participants in the focus group said: *"Since most people accept a join request anyway it is best to allow joining by default and make it possible to dismiss a person that joined later if this is undesirable"*.

| **Lesson Learned 8** |
|---|
| An implicit join process is preferable to an explicit join process |



| | Acquiring_Attention | Notifying_Others | Dismissing_Participants | Dismissing_Viewers | Listening | Joining | Inviting |
|---|---|---|---|---|---|---|---|
| Acquiring_Attention | | | | | | | |
| Notifying_Others | > | | | | | | |
| Dismissing_Participants | x | x | | | | | |
| Dismissing_Viewers | x | x | x | | | | |
| Listening | > | x | > | > | | | |
| Joining | > | > | > | > | x | | |
| Inviting | > | > | > | > | x | x | |

**Table 6.4:** *Comparative Analysis - How well Communico supports the actions to be carried out on a conversation*

Additionally, by making it possible to explicitly dismiss participants the original participants of the conversation still have the option to ask the newly joined participant to leave. In the previous chapter *Joining* and *Inviting* are rated as most important so even though their realization is rated best here as well (see table 6.4) it makes sense that focus group participants propose improvements to the joining process. It is also worth mentioning that, in the previous chapter, we concluded adding participants to a conversation is more important than removing people. So, the fact that focus group participants specifically proposed to add an explicit dismiss participant option indicates that removing participants is in fact important, even if it is less important than adding participants to a conversation. Finally, the focus group participants also suggested to support explicitly notifying people outside the conversation of a conversation that might be interesting to them and to include something like an attention buzzer to acquire the attention of the other participants in the conversation.

### 6.6.4 Benefits and Challenges of Finished Conversations

#### Benefits

The benefits of having access to finished conversations we identified in chapter 5 are the following:

| | |
|---|---|
| • Having access to knowledge you might otherwise forget | `Own Knowledge` |
| • Access to technical knowledge of colleagues | `Technical Knowledge` |
| • Acquiring involvement with your colleagues | `Involvement` |
| • Enjoying your work | `Enjoying` |
| • Acquiring insight in the communication structure | `Communication Structure` |

Having access to your own knowledge was rated as exploited best (see table 6.5). This viewpoint was shared by the participants of the focus group and interviewees. However, like for the benefits of overhearing active conversations, the (dislocated) interviewees voiced the opinion that *Involvement* is also exploited particularly well.

| **Lesson Learned 9** |
|---|
| Having access to finished conversations results in a stronger increase of involvement with colleagues for people that work more dislocated from their colleagues than for those that work more co-located |

| | Enjoying | Communication_Structure | Involvement | Technical_Knowledge | Own_Knowledge |
|---|---|---|---|---|---|
| Enjoying | | | | | |
| Communication_Structure | > | | | | |
| Involvement | > | x | | | |
| Technical_Knowledge | > | x | > | | |
| Own_Knowledge | > | > | > | > | |

**Table 6.5:** *Comparative Analysis - How well Communico exploits the benefits of having access to finished conversations*

The improvements of Communico identified with respect to these benefits were mostly similar to those identified for the benefits of overhearing active conversations. However, because the list of finished conversations is considerable longer than the list of active conversations the focus group suggested the addition of a date range filter to make this list more manageable. In the previous chapter we also suggested it is particularly important to find ways to extract data from the set of finished conversations because having access to your *Own Knowledge* and to *Technical Knowledge* were the two benefits of having access to finished conversations that were found to be most important.

### Challenges

The challenges of having access to the finished conversations we identified in chapter 5 are the following:

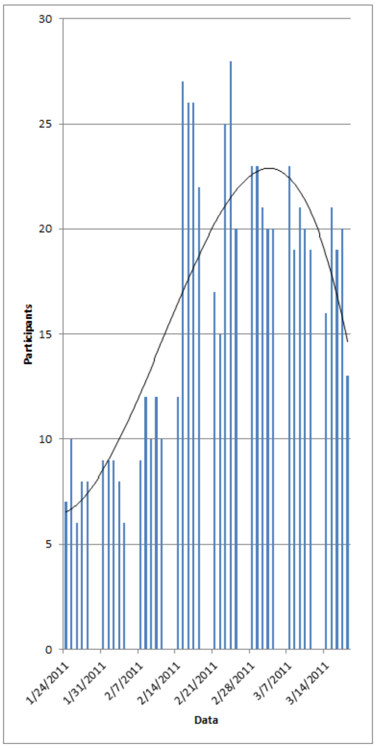
| | |
|---|---|
| • It can be distracting from the current work activities | `Distracting` |
| • The context of the conversation can be unclear | `Context` |
| • A lack of control for the people whose conversations are overheard | `Lack Of Control` |

In the focus group and interviews the same limitations and possible improvements of Communico were discussed as for the challenges of overhearing active conversations with the addition that participants suggested alleviating *Lack Of Control* by making it possible to make conversations private after they finish because people often forgot to do this.

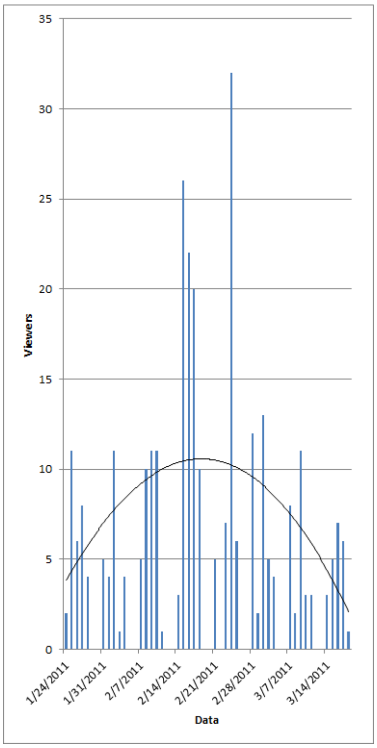
| **Lesson Learned 10** |
|---|
| Only enabling making conversations private before they finish provides insufficient control for the participants of the conversation |

### 6.6.5 Behavior Analysis

To be able to analyze how the users of Communico interacted with the system we used automatically generated transaction logs. From these logs we derived that in the total four months of usage 53 unique users used Communico for 4185 hours in total. The average length of a user session was about 4.4 hours. The users had 1921 conversations in total. The average number of participants of a conversation was 2.2 and the highest number of participants in a conversation was 9. During this period 605 view actions took place at 493 different conversations, the average number of viewers of a conversation was 0.31 and the highest number of viewers was 5.



**(a)** *Number of users*



**(b)** *Number of view actions per day*

**Figure 6.7:** *Behavior analysis*

We divided the total period of four months of usage of Communico in two parts of two months. In the first two months we deployed Communico to a select number of people to test the usage of Communico in the specific setting of Exact, resolve problems and adapt Communico to best fit the needs of this specific setting. In the subsequent two month period we made Communico available to the entire department on a voluntary basis. In the period Communico was available to the entire department we identified a trend in the use of the tool. After we made the tool available, the usage increased, peaked and subsequently decreased when we stopped actively promoting its use. This can be seen in figure 6.7 where we show the increase and subsequent decrease in total number of participants and total number of view actions per day for all users of Communico. Especially in the second graph we see a strong decrease in the number of view actions while in the other graph we also see a decrease of use near the end following a peak.

| **Lesson Learned 11** |
|---|
| In settings where part of the team works co-located the use of technology, supporting the overhearing of conversations without offering specific advantages over the co-located setting, will strongly decrease over time. |

It is also interesting to see the relatively large reduction in number of view actions relative to the reduction in participants.

| **Lesson Learned 12** |
|---|
| A decrease in the number of participants of a tool supporting the overhearing of conversations will result in a stronger decrease in the number of overhear actions |

## 6.7 Discussion

In this section we will reflect on the findings and discuss the most important results. Before this discussion it is important to re-emphasize the context in which this discussion takes place: a practical case setting in which we supported the overhearing of conversations. Outside of this scope several of the lessons learned we have introduced in the previous section and discuss in this section, have already been published. For example in relation to Lesson Learned 2, Sarma et al. [Sar09] report on their tool Tesseract which shows the social network of software engineers as determined by their communication records, which in their case concerned email communication, comments about a bug and work performed and submitted in the bug tracker. Another example, in relation with Lesson Learned 11, is the work of Venolia et al. [Ven10] who are particularly interested in supporting distributed teams that are mostly collocated except for one remote member. In the rest of the discussion we focus on the value of overhearing conversations in global software engineering.

To start, we found that, when deploying awareness sharing technology into settings where part of the team works co-located, the use of this technology should offer specific advantages over the co-located setting to stimulate the use by people that work mostly co-located. This stimulation is required because of a combination of three factors. Firstly, we found in the use of Communico that on the one hand, the people working mostly distributed value the overhearing of conversations with Communico most since it gives them access to information they did not have access to before. On the other hand however, the people working mostly co-located value the overhearing of conversations less because these people already overhear a significant portion of the conversations outside of Communico because they can communicate face-to-face (Lesson Learned 1). Therefore the mostly co-located people have less incentive to use the tool. Secondly however, it is necessary the mostly co-located people also use Communico to actually make the technological support for overhearing conversations work. Since without their cooperation, their conversations are still inaccessible to and cannot be overheard by, the distributed people. Finally, we also identified a strong decrease in the use of Communico at the end of the study, when we were no longer present daily, indicating that our presence artificially stimulated the use.

Related to the previous finding we found that to stimulate acceptance, and therefore use, of awareness sharing technology, it is important to provide the users with more control over the information about them which is being shared (this includes their actions). This can also be seen in some of the lessons learned discussed in the previous section. Firstly, people propose to enable making conversations private before a conversation starts (Lesson learned 5) and after it finishes (Lesson Learned 10). Next to this, people also propose to not show private conversations in the active conversations list to non-participants (Lessons Learned 6).

Another important result we identified is that being able to properly detect interesting conversations is essential. In a traditional office setting this happens unobtrusively and it is important to approach this standard when constructing technological support to enable the overhearing of conversations. This can also be seen in the ideas for future improvements identified is this chapter. Examples of this are being able to automatically identify the subject of and relations between conversations (Lessons Learned 7 and 4). Related to this, also the prevention of information overload is important. When we are better capable of detecting when a conversation is interesting, we will also be better at preventing people from getting too much information (Lesson Learned 3). An additional factor in determining how much information and what granularity of information is needed has to do with the current activity of the user. An example mentioned in the focus group is to put on *'virtual head phones'* to withdraw yourself from the open conversation space and avoid disruption when performing a task that requires significant attention.

## 6.8 Threats to Validity

A threat to external validity is that we only studied one department in a single company. To be able to generalize the findings the study should be repeated in more settings. Next to this, also the size of the sample is a threat to external validity. For practical reasons, we performed one focus group and sent the questionnaire to 47 people of which 25 reported having used Communico actively. Also the sample we selected for the focus group may not be representative as well since we selected all participants from the Delft location for practical reasons. To mitigate this risk we performed semi-structured interviews with the dislocated people.

There are also threats to internal validity. Firstly, most people worked from the Delft office on a daily basis. However, most participants were used to working in a distributed setting, often worked from home and there were three participants from Belgium and two from the USA as well. It is possible that the limited exposure of a portion of the participants to working distributed from their colleagues caused items to be misrated in the questionnaire. Next to this, the people that participated in the focus group and semi-structured interviews also participated in the subsequent questionnaire. This could have biased the results due to a learning effect caused by repeated testing.

We attempted to mitigate threats to reliability by rigorously describing our research site and methods and making all of our quantitative data available online. We do this in an attempt to make, both our data gathering methods and the analysis of our data, repeatable. Subsequently, a threat to construct validity is mono-operation bias. Because we only researched supporting the overhearing of conversations with technology with one specific tool one could argue the results only apply to the use of that tool. We mitigated this threat by defining an explicit set of requirements of open conversations spaces and discussing how Communico fulfills these.

Finally, there is also a threat to statistical conclusion validity. In our analysis of the survey data we used Fishers' Least Significant Difference method to help reduce the number of false positives caused by the pair-wise comparison of all items. In comparison with other methods which aim to accomplish this, Fishers' LSD is fairly liberal. We chose to use a fairly liberal method because of the exploratory character of this research.

## 6.9 Concluding Remarks

### 6.9.1 Conclusions

In this chapter we presented a technological implementation which enables the overhearing of conversations in a distributed setting: Communico. Subsequently we explained how this solution fulfills the requirements of an open conversation space. Next, we performed an empirical case study to measure the value of overhearing conversations in global software engineering from actual industrial experience. This study resulted in the following contributions: (i) indications of how well the benefits and challenges of having access to active conversations are exploited and alleviated, (ii) indications of how well conversations are represented in the open conversation space, (iii) indications of how well actions on a conversation are supported, and (iv) indications of how well the benefits and challenges of having access to finished conversations are exploited and alleviated. Finally, we concluded that overhearing conversations is valuable to global software engineering teams. The most important results of this chapter are:

- The value of awareness sharing technology is higher for people that work more distributed from their colleagues
- The value of awareness sharing technology is higher when a larger portion of a team uses them
- In settings where part of the team works co-located the use of awareness sharing technology should offer specific advantages over the co-located setting to stimulate the use by people that work co-located
- The acceptance, and therefore the value, of awareness sharing technology can be increased by providing the users with more control over the information about them which is being shared
- The value of technological support for overhearing conversations in distributed settings is higher when such support can more accurately detect interesting conversations while preventing an overload of information

Future work will concern the investigation of how to deal with settings in which a portion of the team is distributed while another portion mainly works co-located. We feel this is not only essential to the success of a tool supporting the overhearing of conversations but for awareness sharing technology in general.

### 6.9.2 Virtual Office Implications

In this chapter we have reported on an industrial evaluation of being able to overhear conversations in a distributed software engineering team. As such this chapter directly contributes to answering the second research question of this dissertation, namely:

**Research Question 2**

*What is the value of overhearing conversations in global software engineering?*

From the contributions and the lessons learned identified in this chapter we could not derive new requirements of a virtual office. However, we could make recommendations on how to implement the requirements related to communicating in a virtual office. We identified the following recommendations:

- Only showing the conversations is not sufficient to acquire insight in the communication structure of the team
- It is important to have sufficient control over both the amount conversations you overhear, and the conversations in which you participate
- An implicit join process is preferable to an explicit join process.
- Making the relations between conversations clearer improves the clarity of the context
- Deciding whether a conversation is interesting can be done quicker and less obtrusive when its subject is known

By presenting the value of overhearing conversations in global software engineering teams we answered the third and final part of the second research question of this dissertation. The first part of this question was answered in chapter 4, in this chapter we provided a theoretical motivation why the overhearing of conversations is valuable. The second part of this research question was answered in chapter 5, in which we evaluated the value of overhearing conversations in the field of software engineering.

Chapter 7

# An Industrial Evaluation of Mood-Based Microblogging

*Distributed teams face the challenge of staying connected. How do team members stay connected when they no longer see each other on a daily basis? What should be done when there is no coffee corner to share your latest exploits? In this chapter we evaluate a microblogging system which makes this possible in a distributed setting. The system, WeHomer, enables the sharing of information and corresponding emotions in a fully distributed organization. We analyzed the content of over a year of usage data by 19 team members in a structured fashion, performed 5 semi-structured interviews and report our findings in this chapter. We draw conclusions about the topics shared, the impact on software teams and the impact of distribution and team composition. Main findings include an increase in team-connectedness and easier access to information that is traditionally harder to consistently acquire.*

This chapter is strongly based on our publication *"Fixing the 'out of sight out of mind' problem - One Year of Mood-Based Microblogging in a Distributed Software Team"* in the Proceedings of the 10th International Workshop on Mining Software Repositories (MSR 2013). This publication is co-authored by Kevin Dullemond, Margaret-Anne Storey and Arie van Deursen.

## 7.1 Introduction

The year is 2011 and in a young and small software company called IHomer a significant problem is being discussed. Because the default work location in the company is home the people work distributed from each other most of the time, and have always physically met up once a week on Tuesdays in order stay connected. However, one employee has been away on contract for a prolonged period now, preventing him from attending the weekly meet-ups and he is starting to feel more and more disconnected from the rest of the team. On the one hand he is starting to lose touch with what occupies his colleagues and on the other hand his colleagues weren't even aware he was feeling unhappy with the overall situation. Fast forward almost two years to the present day and the issue has been dealt with by the introduction of WeHomer, a microblogging environment in which people at IHomer share their activities and moods with each other, to stay current on each other's feelings, experiences and latest exploits and as a result stay connected as a team.

The story above is not unique as software engineering is becoming more and more distributed. This is caused by the increasing globalization of business [Car99, Her01, Her07] and the rising popularity of working from home [Die09]. At the same time, significant challenges are faced when collaborating in a distributed setting as reported in the well-known work of Olson and Olson [Ols00]. More recently Nguyen et al. [Ngu08] have investigated whether the effects of distance on distributed communication delay and task completion reported in the literature still exist. They report that advances have been made but also that more work is needed and there are still many open research questions.

We believe both microblogging and mood sharing are essential to alleviate challenges arising from this distributed nature. On the one hand microblogging is essential, since being able to exchange small elements of content makes people feel more connected with others, especially when people work distributed from their colleagues [Zha09, Zha10, Ehr10]. On the other hand mood sharing is essential, since being aware of the emotional state of your colleagues makes it possible to act accordingly and achieve better results in joint work [Gar99]. Therefore, in this research, we aim to determine whether an environment in which one can both express himself and get a sense of how his team members feel is valuable to distributed software engineers. In this chapter, we use a microblogging solution extended with mood indicators (MBMI) called WeHomer to learn from the use of such an environment. The main goal of the chapter is:

*"To understand how microblogging with mood-indicators helps distributed organizations in knowledge sharing"*

Furthermore we have identified four research questions:

- What sort of topics are discussed in MBMI?
- What is the impact of the introduction of a MBMI on a software team?

- What is the impact of distribution on the use of a MBMI?
- How does team composition impact collaboration with MBMIs?

We structure this chapter as follows: In section 7.2 we discuss related work to this research. Next, in section 7.3 we discuss the research site and methods of data collection and analysis. Subsequently, in section 7.4, we show descriptive statistics and in section 7.5 we present the most important findings. Finally we discuss threats to the validity of our study in section 7.6 and conclude upon our work and discuss future work in section 7.7.

## 7.2 Related Work

Software engineering is by nature a highly collaborative activity, and having access to knowledge about the context in which you are working is essential to properly collaborate with others [Sch02, Syr97]. In literature this kind of knowledge is commonly referred to as *'awareness'* [Dou92, Sch02]. When working co-located this information is exchanged relatively passively and unobtrusively [Sch02, Fog05], so people are continuously aware of information related to their current context [Dul10]. However, when people no longer share a physical work environment exchanging information without technological support becomes unfeasible [Dul10]. Therefore, the (global) software engineering community has developed and studied a wide variety of tools, for example: Instant Messaging, email, issue management systems and configuration management systems.

According to Bly et al. [Bly93] it is particularly important to recognize the need for informal interactions, spontaneous conversations, and general awareness of people and events when teams are geographically dispersed. These informal non-work related conversations and events are powerful enough to facilitate the emergence of trust [Wan13]. Storey et al. advocate further research on understanding how social media plays a role in (global) software engineering [Sto10]. One of the potential implications they identified in their research, concerns the challenges which arise when teams are distributed across time zones and geographical locations, and lack informal mechanisms for communication. They emphasize social media is regarded as a mechanism that can support informal and serendipitous interactions across the team, and as such can alleviate these challenges. Finally, they present ten research questions, of which the following three are applicable to this study:

1. Can social media play an effective role in supporting coordination and task articulation?
2. What kinds of social media would increase informal communication, the flow of knowledge and awareness across team and project members?

3. What are the drawbacks from increased transparency in team projects? Does this lead to privacy concerns?

There are several social media tools available to software engineers which facilitate coordination, communication with and learning from other users, being informed about new developments and creation of informal documentation [Sto10]. These tools can be characterized by an underlying *'architecture of participation'*; systems that are designed for user contribution [O'R04]. Such a design supports the creation of collective value, often as an automatic byproduct of an individual activity. Wikis, blogs and microblogs are some well-known examples of such social media solutions.

In our research we aim to determine whether an environment in which one can both express himself and get a sense of how his team members feel is valuable to distributed software engineering teams. The specific environment we used, called WeHomer, allows users to exchange small elements of content such as sentences, images and hyperlinks, and their corresponding emotion. In fact, we use a microblogging solution extended with mood sharing functionality. Several research projects have been conducted in the field of software engineering to increase the understanding of how and why people use microblogging solutions. We will consider three of these user studies, namely studies on Twitter [Zha09], Yammer [Zha10] and BlueTwit [Ehr10], to identify similarities and differences between these microblogging solutions and WeHomer.

Firstly, Twitter is a publicly available microblogging service with which users can publicly share messages, limited to 140 characters. They can also indicate whether their messages are public or private. When messages are indicated to be public they are accessible to all users of Twitter. However, when messages are indicated to be private they are only accessible to those users who have subscribed and are explicitly authorized to the user his feed. Zhao and Rosson [Zha09] identified several characteristics in the use of Twitter. Important examples are: (i) frequent small updates of personal life events enable users to stay aware of people they do not encounter on a daily basis and (ii) subscribing to people you personally know and selected enables users to get trustworthy and useful information.

Secondly, Yammer[1] is very similar to Twitter, the main difference being that Twitter is publicly available while Yammer is enclosed by organizational boundaries. Other differences are the absence of a character limit, the possibility to create private and public groups and the opportunity to add attachments to messages. Zhang et al. [Zha10] also identified several characteristics of the use of Yammer. They give an indication that users use Yammer more for publishing news about their groups or business units than for news about themselves. Subsequently, they indicated that Yammer was used to have long conversations and discussions. Finally, they found that Yammer enables users to stay aware about what others are working on and to make new connections.

[1]http://yammer.com

BlueTwit is also very similar to Twitter, however it differs on two points: (i) BlueTwit is only available within organizational boundaries and (ii) BlueTwit has a character limit of 250 instead of the 140 character limit of Twitter [Ehr10]. Ehrlich and Shami [Ehr10] discussed characteristics of the use of BluetTwit, It enables: (i) having internal conversations about confidential information, (ii) staying aware of what others are working on and (iii) enhancing your reputation.

All of the above user studies on the usage of different microblogging solutions mentioned an important side effect of microblogging in general: people feel more connected with each other. This is especially the case when people work distributed from their colleagues since microblogging kept them connected to other colleagues and the company, and alleviated the feeling of isolation. WeHomer differs from BlueTwit in the sense that it drops the character limit, enforces subscription to all users automatically (manageable because of the small community), and adds mood sharing functionality.

Garcia et al. [Gar99] introduced *'Emotional Awareness'* and argue that it enables users to become aware of the emotional state of their collaborators and act accordingly to achieve better results in their joint work. Other research mainly focuses on electronic meeting systems in which each participant explicitly specifies his mood and changes in the average mood are visualized to all participants [Mor11, Fes12]. WeHomer integrates both microblogging functionality and the opportunity to express your current mood into a single environment. This is the main differentiator of WeHomer in comparison with other microblogging solutions.

## 7.3 Research Site and Method

### 7.3.1 Research Site

This research is carried out at IHomer, a Dutch software engineering company founded in August of 2008 in which it is common practice to work from home. At the time of this study IHomer employed 20 people, working on a variety of products, projects and contracts. The largest team consists of 7 people working on related projects, but the overall group is very close with personnel moving between teams and teams exchanging projects as needed. Even though it is common practice in the company to work from home, the employees try to get together once a week on Tuesdays to meet face-to-face at an office to stay connected. Sometimes this can be difficult however, for example when someone is away on a contract and has other obligations on Tuesdays. The company has grown over the past years and initially on Tuesdays everyone discussed what they were doing. This worked well until the company size reached 16, and then sub teams were formed to keep this face-to-face communication more tractable. Teams cluster according to various factors: projects and related technologies being two of them.

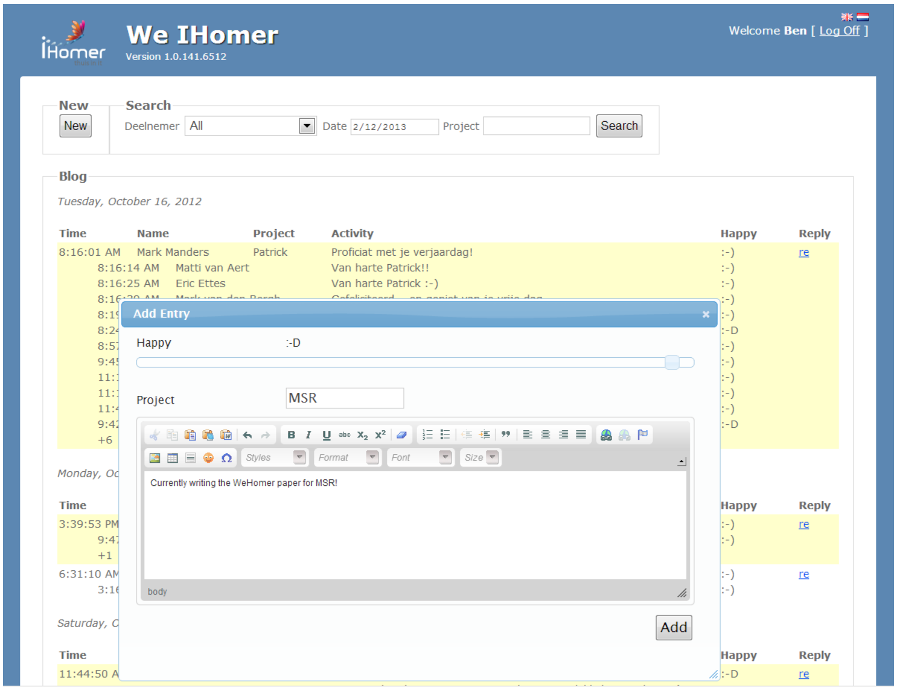
People at IHomer aim to work together closely and stay a very connected community. One of the core strategies to stay connected is the weekly face-to-face meeting. As mentioned above this is not always feasible which can become a practical problem if people are unable to attend the majority of the Tuesday meet-ups for a prolonged period. In order to cope with these issues the WeHomer system was developed by an employee of the company (not an author of this paper) and deployed in October 2011. It is a platform on which IHomers can share information about their day with their colleagues in order to stay connected and increase awareness. Users can share information about a new topic, called an entry or respond to an existing topic, called a comment (commenting was not supported in the first three months of our data analysis period). Comments are shown in chronological order grouped together under the entry to which they correspond. We use the term post to refer to something that is either an entry or a comment. Posts cover such items as what you are doing right now, what you have done, what you are going to do, how you feel about something and random thoughts.

Associated with each post is a happiness score ranging from 0 (totally unhappy) through 100 (utter bliss) depicting how the user feels about this post. In the user interface of WeHomer (see figure 7.1) the happiness index can be selected by use of a slider bar which shows one of 5 discrete emoticons corresponding to the level that is selected by the user[2]. The exact integer value of the happiness index is not derivable by end users.

### 7.3.2 Method

The primary method of data collection we use is mining the WeHomer data between October 2011 and November 2012 by analyzing and subsequently coding the content. During this period there were a total of 1312 entries and 1189 comments. Because it is feasible to hand code each of these entries and comments sampling was unnecessary and we analyzed the content of all entries and comments. The coding was done by the first two authors and the coding set was arrived at in an iterative fashion. Firstly, a random (but consecutive) sample of 50 entries (including the corresponding comments) was selected and coded independently by both coders. Following this the two coders compared their codes and discussed their reasoning behind those codes. Based on this discussion the coders agreed on a joint coding set with which they independently coded another random (consecutive) sample of 25 entries (again including the corresponding comments) and discussed discrepancies in how they coded the sample. Based on this discussion they refined the coding set and did another iteration. After a total of three such iterations they decided the coding set was consistent between them and they could go ahead with the actual coding. To do this they divided the total data set in six ranges of approximately 200 entries and each coded three non-consecutive ranges.

[2]Happiness index ranges: '>:-(' = [0,20), ':-(' = [20, 40), ':-|' = [40, 60], ':-)' = (60, 80], ':-D' = (80, 100]



Deelnemer All Date 2/12/2013 Project

Blog

*Tuesday, October 16, 2012*

| Time | Name | Project | Activity | Happy | Reply |
|---|---|---|---|---|---|
| 8:16:01 AM | Mark Manders | Patrick | Proficiat met je verjaardag! | :-) | re |
| 8:16:14 AM | Matti van Aert | | Van harte Patrick!! | :-) | |
| 8:16:25 AM | Eric Ettes | | Van harte Patrick :-) | :-) | |
| 8:16:30 AM | Mark van den Bergh | | Gefeliciteerd ... en geniet van je vrije dag | :-) | |
| 8:1 | | | | :-) | |
| 8:2 | | | | :-D | |
| 8:5 | | | | :-) | |
| 9:4 | | | | :-) | |
| 11: | | | | :-) | |
| 11: | | | | :-) | |
| 11: | | | | :-) | |
| 9:4 | | | | :-D | |
| +6 | | | | | |

*Monday, Oc*

| Time | Happy | Reply |
|---|---|---|
| 3:39:53 PM | :-) | re |
| 9:4 | :-) | |
| +1 | | |
| 6:31:10 AM | :-) | re |
| 3:1 | :-) | |

*Saturday, C*

| Time | Happy | Reply |
|---|---|---|
| 11:44:50 A | :-D | re |

Happy :-D

Project MSR

Currently writing the WeHomer paper for MSR!

Figure 7.1: *WeHomer user interface*

Subsequently, based on what we found in the content analysis we conducted semi-structured interviews (see Appendix E for the interview structure) with five of the nineteen users in which we asked questions about what was unclear to us in the analysis and follow-up questions we had based on the analysis. To select which five people to interview we used purposive sampling in order to get an as complete view as possible. In the selection process we explicitly excluded authors of this paper. We did select the original developer of the WeHomer system, the person with the highest number of posts, the one with the lowest number of posts and the two people with respectively the highest and lowest number of entries to number of posts ratios.

## 7.4 Descriptive Statistics

In this section we present information derived from the mined data to present the reader with an image of how the MBMI environment is used.

In figure 7.2 the weekly average number of entries, comments and posts is shown. In this figure it can there is significant and consistent use of the WeHomer system for the entire year we are investigating. Additionally we can see that at the start of the period there were no comments and considerably more entries than the remainder of the period. This is because at the start posting comments was not possible and people used entries to comment on another entry by referring to the entry they wished to comment on.

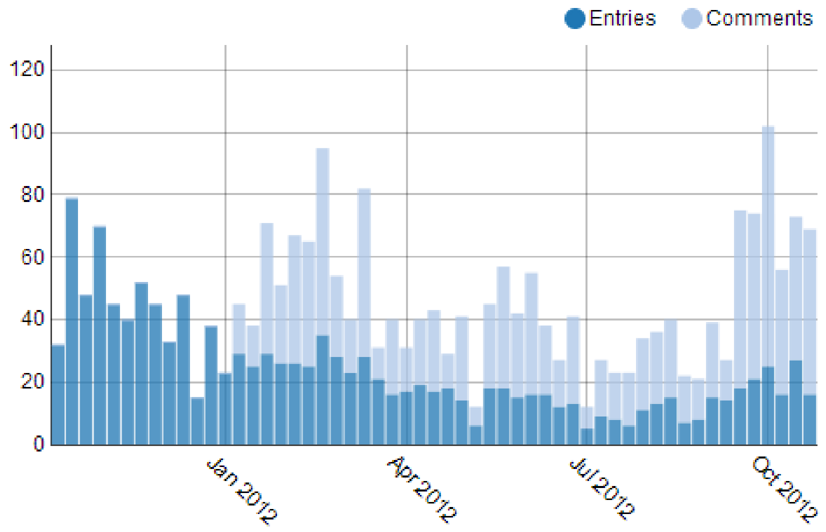


**Figure 7.2:** *The number of entries, comments and total posts per week*

Subsequently, in table 7.1 we present the median length and the interquartile range of entries, comments and for entries and comments combined to give an indication of their respective lengths. So, for instance, we see that 75% of the entries are shorter than 143 characters.

| | Lower Quartile | Median | Upper Quartile |
|---|---|---|---|
| Entries | 62 | 97 | 143 |
| Comments | 22 | 49 | 86 |
| Posts | 39 | 75 | 122 |

**Table 7.1:** *Average length in characters*

In figure 7.3 for the entire year the average happiness per week as well as the maximum and minimum happiness score are shown. In this figure it can be seen that the happiness fluctuates significantly and that in general the highest and lowest score for a week lie relatively far apart.

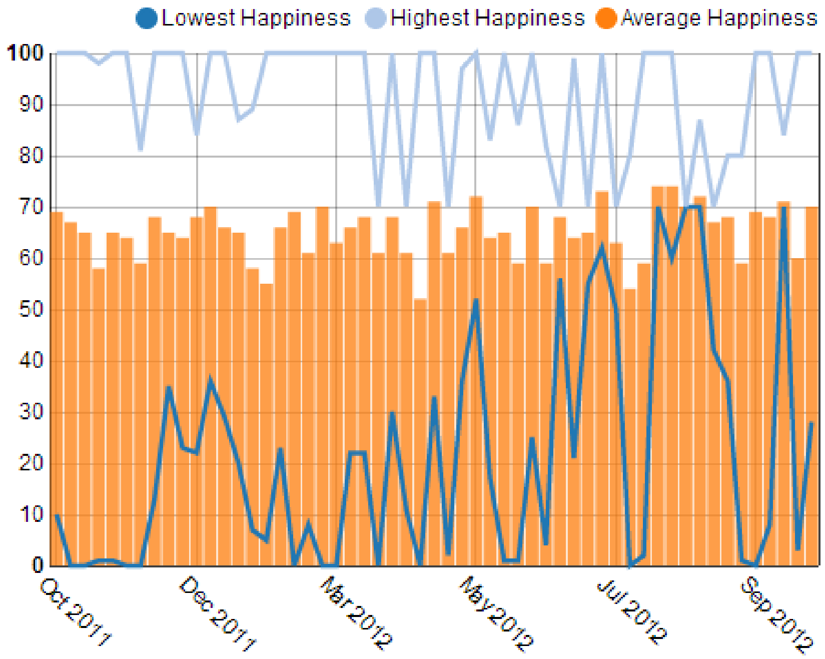


**Figure 7.3:** *The average, lowest and highest happiness score per week*

Further, in WeHomer a default happiness score is automatically selected for each post a user makes, namely 70 on the range between 0 and 100. If a user doesn't manually select another happiness score this default score is used. Therefore it is interesting to investigate how often the users deviated from this default value. In table 7.2 we show separately for entries and comments how often this occurred. In this table we can see that a significant portion of the happiness scores were set at the default score. Therefore, we asked the five people we interviewed whether these values were chosen consciously. They all told us that even though they on occasion forgot to change the happiness score, in general they spent a minute to think which score to select, even if this is the default happiness score.

| | Default | | Non-Default | | Total | |
|---|---|---|---|---|---|---|
| Entries | 791 | 60.3% | 521 | 39.7% | 1312 | 52.5% |
| Comments | 931 | 78.3% | 258 | 21.7% | 1189 | 47.5% |
| Total | 1722 | 68.9% | 779 | 31.1% | 2501 | 100% |

**Table 7.2:** *Percentage of posts with the default happiness index*

Finally, in IHomer there exist 4 teams of people working on related projects.

We compare the amount of directed communication between members of the same team and members of different teams to give an indication of how much collaboration was being done overarching the different teams. To do this we do the following: Firstly, we define commenting on an entry posted by a specific user as the utilization of a directed communication line between those two users. We do this even though all other users can see this communication because the communication is at least directed at that specific user. Subsequently we sum all of these utilizations of communication lines. An interactive visualization of the utilization of communication lines between both team members and non-team members can be seen at `http://aspic.nl/msr2013/vis/sna`.

Following this we needed to compare the amount of communication within teams with the amount of communication outside of teams. We did this by doing the following for each team:

1. Count the total number of comments of the people in the current team on other people within the current team
2. Count the total number of comments of the people in the current team with the people outside the current team
3. Make the number found in step 1 relative to the team size (n) by dividing it by n*(n-1) (n*(n-1) to represent the total number of communication lines)
4. Make the number found in step 2 relative to the number of people outside of the team

A summary of these results is shown in table 7.3:

| **Team** | **Step 1** | **Step 2** | **Step 3** | **Step 4** |
|---|---:|---:|---:|---:|
| Team 1 (5 people) | 49 | 162 | 2.45 | 0.89 |
| Team 2 (7 people) | 386 | 194 | 9.19 | 1.47 |
| Team 3 (5 people) | 47 | 161 | 2.35 | 0.88 |
| Team 4 (2 people) | 24 | 165 | 12.00 | 0.61 |

**Table 7.3:** *Utilization of Communication lines within and between teams*

In this table it can be seen that the utilization of communication lines within teams is considerably higher than communication lines crossing team boundaries.

## 7.5 Findings

In this section we answer the research questions specified in section 7.1. We structure the section based on the research questions, answering each research question in a separate subsection.

### 7.5.1 Topics

Research question 1 is: *"What sort of topics are discussed in MBMI?"* We answer this research question by first discussing the set of codings we used to code the data to give insight in the variety of topics discussed in the MBMI system. Subsequently we show the occurrence of each of the codings in both the entries and comments to give insight in the frequency each topic occurred. Finally we analyze these occurrences and make generalizations.

As described in section 7.3 we used an iterative bootstrapping process to construct the coding set. This coding set is structured in four major categories:

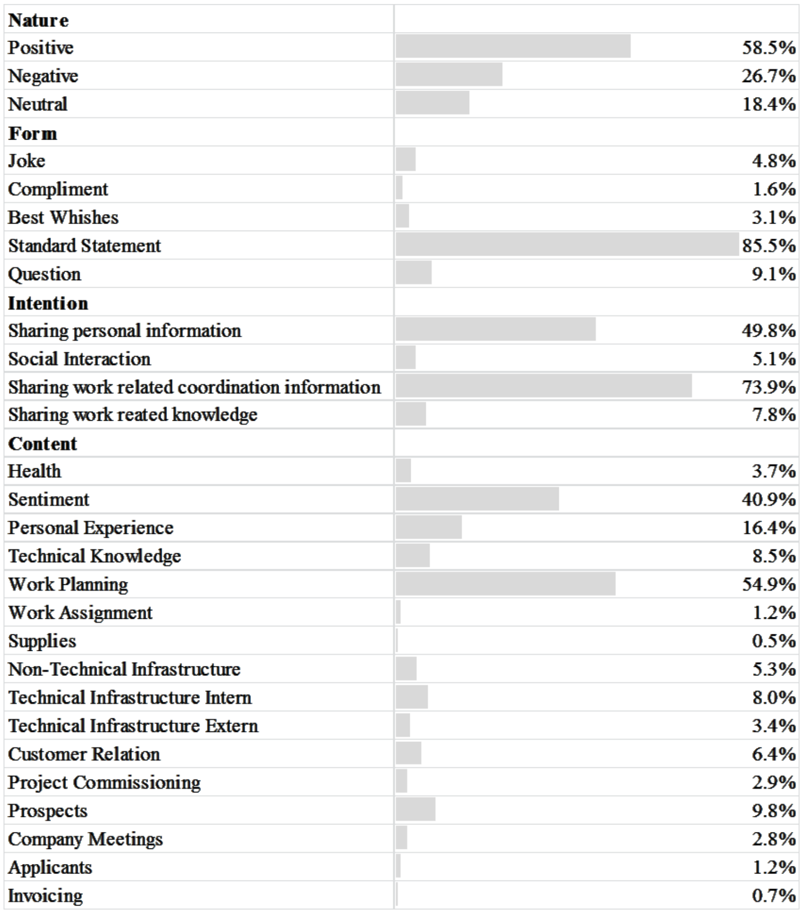
1. Nature
2. Form
3. Intention
4. Content

Each of these coding categories is further divided into sub-categories and actual codes. In appendix F we show this subdivision for each of the four major coding categories depicted as trees. In these trees leaf nodes depict actual codes while non-leaf nodes depict sub-categories. In the appendix we also explain each of the codes and give examples.

Subsequently we applied all four coding categories[3] to the entries and the first two coding categories to the comments. We only apply codes of the first two categories to the comments because comments are made in the context of an entry which makes it difficult to differentiate in how far the intention and content of a comment are dictated by the corresponding entry. We present the occurrence of each of the codes in the set of entries and the set of comments in figure 7.4 and figure 7.5 respectively.

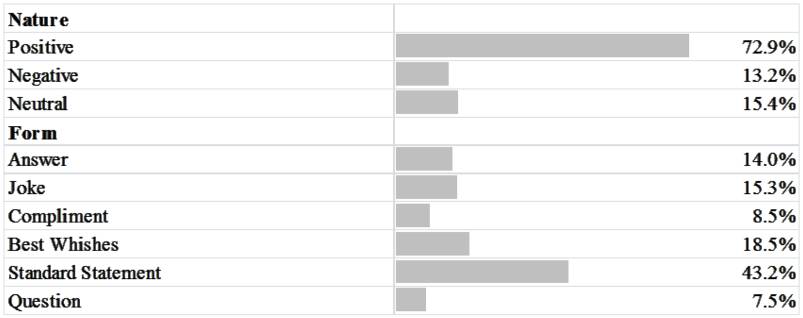
In figure 7.4 it can be seen the most frequently occurring codes are *'Statement'* (85.5%), *'Coordination'* (73.9%), *'Positive'* (58.5%), *'Work Planning'* (54.9%) and *'Personal Information'* (49.8%). In addition to this, four of the five people we interviewed indicated they have the tendency to post more positive things and to post more when they are in a positive mood. Further the consensus in the interviews about what they post is that it is everything they consider useful or interesting to their team members. One would suspect this to lead to a diverse list of topics to be shared on the medium and both the diversity of codes and the relatively distributed occurrence of these codes support this expectation. Finally, when asked to specify what they shared most, popular subjects mentioned are: personal information, project information with the intention to coordinate, technical information and prospects. This also corresponds with the actual data presented in figure 7.4.

[3]Note that the codes in the four categories are not mutually exclusive in their application to posts (e.g. a post can contain a positive and negative part)



| | |
|---|---|
| **Nature** | |
| Positive | 58.5% |
| Negative | 26.7% |
| Neutral | 18.4% |
| **Form** | |
| Joke | 4.8% |
| Compliment | 1.6% |
| Best Whishes | 3.1% |
| Standard Statement | 85.5% |
| Question | 9.1% |
| **Intention** | |
| Sharing personal information | 49.8% |
| Social Interaction | 5.1% |
| Sharing work related coordination information | 73.9% |
| Sharing work reated knowledge | 7.8% |
| **Content** | |
| Health | 3.7% |
| Sentiment | 40.9% |
| Personal Experience | 16.4% |
| Technical Knowledge | 8.5% |
| Work Planning | 54.9% |
| Work Assignment | 1.2% |
| Supplies | 0.5% |
| Non-Technical Infrastructure | 5.3% |
| Technical Infrastructure Intern | 8.0% |
| Technical Infrastructure Extern | 3.4% |
| Customer Relation | 6.4% |
| Project Commissioning | 2.9% |
| Prospects | 9.8% |
| Company Meetings | 2.8% |
| Applicants | 1.2% |
| Invoicing | 0.7% |

**Figure 7.4:** *The frequencies of the codings for the entries*



| | |
|---|---|
| **Nature** | |
| Positive | 72.9% |
| Negative | 13.2% |
| Neutral | 15.4% |
| **Form** | |
| Answer | 14.0% |
| Joke | 15.3% |
| Compliment | 8.5% |
| Best Whishes | 18.5% |
| Standard Statement | 43.2% |
| Question | 7.5% |

**Figure 7.5:** *The frequencies of the codings for the comments*

Subsequently we discuss a deviation from the expected. An evident application for a MBMI system is asking questions. However, we found a relatively low amount of these. The total number of entries that are questions is 9.1% and the total number of comments that are questions is 7.5%. We compared these numbers to the results found in a study by Erhlich et al. [Ehr10] on the investigation of the usage of Twitter and BlueTwit (an internal proprietary version of Twitter) in an organization for people that use both tools. To compare our number of questions to theirs we summarized the percentages for *"Ask Question"* and *"Directed with Question"* in their result set to yield a total number of questions of 6% for Twitter and 13% for BlueTwit (versus our 7.5% and 9.1% respectively). So, in the environment in our study relatively more questions were asked than in the Ehrlich setting for Twitter use, but less than with BlueTwit.

When asked about the amount of questions in the five interviews the respondents indicated that they asked a relatively low amount of questions on the MBMI system for two reasons. Firstly, the medium is asynchronous which makes it unpredictable when a question will be answered. Secondly, they indicated they usually knew who to contact (or at least knew who knew who to contact) and preferred contacting someone who would know the answer directly over asking it to the entire group. It was also discussed in the interviews the low amount of questions is likely to be specific to companies with a relatively small size, close personal connections and transparency between the team members because in such companies people will more easily know who knows what.

### 7.5.2 Impact on a Software Team

Research question 2 is: *"What is the impact of the introduction of a MBMI on a software team?"* The first main impact we found is that members of software teams feel more connected to each other when they are able to share activities and moods. We base this primarily on information from the interviews. Firstly, three out of the five people we interviewed explicitly reported feeling more connected to their colleagues since they were able to share their moods and activities within the team using the MBMI system. Additionally these interviewees also reported being better able to understand their colleagues since using the environment and two of them reported they felt their colleagues understood them better as well. This finding corresponds to the results of the studies of Zhao and Rosson [Zha09], and Ehrlich and Shami [Ehr10] on microblogging in the work place. Zhao and Rosson [Zha09] conclude:

*"Our results suggest that microblogging may help colleagues to know each other better as persons, that is in addition to professional relationships; this benefit is achieved by staying aware of small details about others' personal lives, interests, and current moods, which in turn creates more opportunities for exchanging acknowledgments and social support, generating new common ground, and creating and sustaining a feeling of connectedness."*

| **Lesson Learned 1** |
|---|
| Distributed software engineers feel more connected to each other when they are able to share activities and mood |

As the second main finding, we found a MBMI makes information that is traditionally harder to consistently acquire more approachable and less volatile. This is based on both the content analysis and the interviews. In the content analysis we found that in particular the coding-categories entrepreneurial tasks (14.5% of all entries) and customer relations (9.3% of all entries) represent a considerable portion of the data. One of the interviewees indicated information about these types of activities is traditionally difficult to consistently gather. For instance information about *"how to build a business"* is often shared in face-to-face communication which makes it difficult to acquire at a later time (you will have to ask or try to recall) and the information is likely to be different from the original.

In the interviews people comment they consider it a strength of the system to be able to share non-time critical information: Information they would like to know about *"eventually, but not necessarily within the next five minutes"*. Before WeHomer communicating this type of information was often postponed until a weekly face-to-face meeting or discarded altogether. One of the interviewees even said he found the system to offer benefits over meeting face-to-face on Tuesdays: *"With WeHomer it is easier to stay current than by meeting people face-to-face on Tuesdays because then you don't get to talk to everyone"*.

| **Lesson Learned 2** |
|---|
| A mood-activity environment makes information that is traditionally harder to consistently acquire more approachable and less volatile |

Finally, we also found a MBMI system facilitates an unobtrusive way to express your personal feelings or thoughts to your colleagues. All of the interviewees mentioned they considered the low threshold the environment offered for sharing information with their colleagues an important strength. One of the interviewees said he felt he could *"share knowledge and emotions like you are co-located"*. We can also see the environment offers a light-weight method to share personal information since over half of the entries (52%) contain personal information.

### 7.5.3 Impact on a Distributed Software Team

Research question 3 is: *"What is the impact of distribution on the use of a MBMI?"* Firstly, we found that people who work co-located with the majority of their team, share less activities and moods with those team members that are non-collocated. In our setting this behavior presented itself as follows: while for the rest of the week the default work location is home, on Tuesdays people at IHomer try to work co-located at a central office as much as possible. However, at times this is unfeasible for specific team members, for instance due to being contracted at a customer location. In practice at least half of the team is present on Tuesdays the vast majority of the time, but it is rare for the entire team to be present. Therefore it is striking to see that on Tuesdays the number of entries and comments is significantly lower than on other days of the week (see figure 7.6). Finally, also in the interviews it was recognized that *"on Tuesdays WeHomer is used very little"*. This is similar to what we reported in chapter 6 on the deployment of a conversation overhearing tool in an industrial setting with a non-homogeneous geographical distribution.

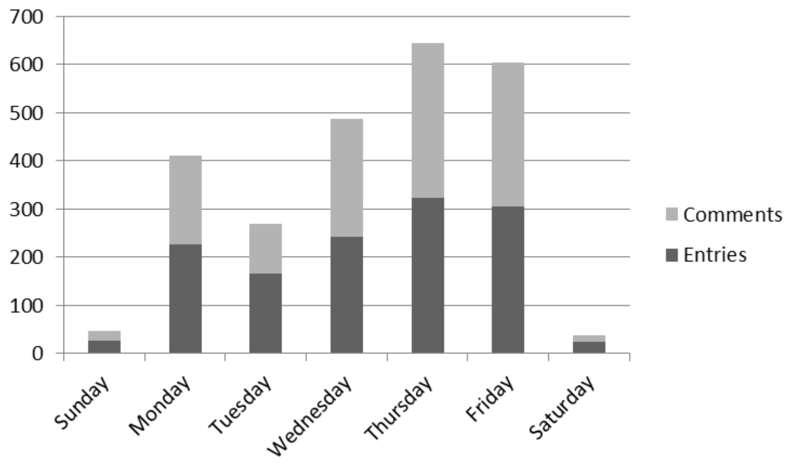


**Figure 7.6:** *The number of entries and comments for each day of the week*

| **Lesson Learned 3** |
|---|
| In distributed software engineering teams, people who work co-located with the majority of their team, share less activities and moods with those team members that are non-collocated. |

Paradoxically to what is discussed above, the interviewees indicated they do find it particularly valuable to share moods and activities with their distributed colleagues. One of the interviewees stated: *"WeHomer is used very little when people work co-located on Tuesdays which makes things less transparent for people that cannot be there"*. The interviewees indicated they recognize the value in making sure the entire team stays connected, even when part of the team works co-located and part of the team works distributed. They recognize the value because they know from experience how difficult being the dislocated colleague can be. It is striking to see that even though they know it is important to help their distributed team members, they still struggle to do so.

| **Lesson Learned 4** |
|---|
| In distributed software engineering teams, people who work co-located with the majority of their team, find it is particularly useful to share activities and moods with those team members that are non-collocated. |

A useful insight on this is also shared by Fullerton of Stack Exchange in his blog post [Ful13] about the lessons learned from three years of working in a distributed team. He states: *"There's no halfsies in a distributed team. If even one person on the team is remote, every single person has to start communicating online. The locus of control and decision making must be outside of the office: no more dropping in to someone's office to chat, no more rounding people up to make a decision. All of that has to be done online even if the remote person isn't around. Otherwise you'll slowly choke off the remote person from any real input on decisions."*

### 7.5.4 Impact of Team Composition on MBMI

Research question 4 is: *"How does team composition impact collaboration with MBMIs?"* Our main finding with respect to this research question concerns the regularity in which the members of a team use the MBMI: do all the members of the team use the MBMI system an equal amount of the time and for the same sort of topics? If this usage differs significantly between different team members the distribution of relevant information in the team will be unbalanced as well. This was the main challenge in the use of the MBMI system that came forward in the interviews. Examples of things interviewees said are: *"The success of WeHomer is dependent on participation"*, *"There is only a challenge for those not using it"* and *"If you don't participate you miss things"*. The challenge is threefold. Firstly, team members using the environment less than their colleagues run the risk of missing things. Secondly, a team member sharing information cannot be certain his team members know about this. Finally, since the MBMI system isn't used for everything, you cannot infer something did not happen because it is not available there. Therefore, to have a complete view of all valuable information about the mood and activities of team members software engineers need to consult other sources as well.

| **Lesson Learned 5** |
|---|
| If the regularity with which team members utilize the mood-activity environment differs, the distribution of relevant information in the team will be unbalanced |

Finally, in the interviews we talked about the type of teams for which they thought a MBMI system would be beneficial. Firstly, to use the system in the same (personal) fashion as it is being used at IHomer the teams need to be sufficiently involved. One of the interviewees said: *"I need to know the people"* while another said that *"teams need to be homogeneous (shared interests, shared work)"*. They also mentioned that as a direct result of this, team size can become an issue but only of it leads to the team members being less involved with each other.

On the other side of things the interviewees did consider a MBMI system beneficial to all companies. One interviewee said: *"every organization needs a WeHomer because even a closed door is a barrier"*. They do believe however that the type of information that is being shared will be connected to the type of organization. For instance one of the interviewees explained that the he considered the high amount of personal information and the diversity of the messages on WeHomer to be tied to the open character of IHomer and that he would expect a more traditional organization to share a larger portion of technical information instead.

## 7.6 Threats to Validity

Threats to external validity can exist at each of the levels of generalization in a study. In our study, a threat to external validity exists in the generalization of the single software engineering team to the population of all distributed software engineering teams. To be able to better generalize beyond the setting we performed the study in, the study should be repeated in other teams as well. With respect to the generalization of the sampled data to the population of IHomer our work is much less threatened. For the interviews we sampled 5 out of the 20 people in the company (25%) and for the content analysis we even coded 100% of the posts in WeHomer for the year of data we investigated.

We attempted to mitigate threats to reliability by elaborately describing our research site and methods and making our coding set and interview design available. Next to this we also make all of our data available in anonymized form and make the tool available upon request. We do this to make both our data gathering methods and the analysis of our data, repeatable.

Subsequently, a threat to construct validity is mono-operation bias. Because we only researched the application of MBMI environments with one specific tool, one could argue the results only apply to the use of that tool. The only mitigation we need to offer for this is the general nature of the tool itself. Basically any tool with which it is possible to share activities and moods in distributed teams will suffice and the WeHomer tool clearly fulfills these requirements. A final threat to construct validity in this study is that both the creation of the coding set and the coding of the posts were done by the first two authors who are also employees in the company at which the study was being performed. The advantage of this is that the researchers possess insight knowledge and can leverage this to code the data more accurately. A disadvantage is that the researchers might not be completely impartial due to their involvement in the setting. Overall, it is our opinion the advantages outweigh the disadvantages.

## 7.7 Concluding Remarks

### 7.7.1 Conclusions

In this chapter we presented an empirical study on microblogging with mood indicators (MBMI) in a distributed software engineering team. We collected the empirical data by mining the WeHomer data between October 2011 and November 2012. WeHomer is a MBMI in which people can share their activities and moods with each other. Subsequently, the content of all entries and comments was analyzed and coded. Based on what we found in this content analysis we interviewed five distinctive users of the system. In these interviews we asked questions about what was unclear to us in the analysis and we asked follow-up questions we had based on this analysis.

The main contributions of this chapter are the answers to the research questions. First, we answered what sort of topics are discussed in microblogging systems with mood indicators (MBMI) by presenting the nature, form, intention and content of the posts in over one year of usage data and presenting the frequency at which these occurred. Based on this data we found that distributed software engineers primarily share positive posts with the intention to either coordinate or provide personal information. Furthermore, when compared to other corporate microblogging solutions we found a relatively low amount of questions.

Subsequently, we have shown there are two major impacts of the introduction of MBMI on a software team. Firstly, team members become more connected. The loss of teamness is a major and unresolved issue in the field of GSE and therefore advances in this area are significant. Secondly, the MBMI system made information that is traditionally harder to consistently access more approachable and less volatile

Further, on the impact of distribution of the software team on the use of MBMI, we found that the way people act when working co-located with the majority of their team is paradoxical to how they think they should act. On the one hand, people share less activities and moods with their distributed colleagues while on the other hand they do recognize the value in staying connected with the rest of the team. It is striking to see that even though they know it is important to help their distributed team members, they still struggle to do so.

Next, on the impact of team composition on collaboration with MBMI, we found that the distribution of relevant information in the team will be unbalanced if team members use the environment unequally.

Concerning future work we are particularly interested in researching the actual value of incorporating mood in microblogging systems. A way to accomplish this is to perform two user studies in similar software teams in which one of the teams receives access to a regular microblogging solution and the other team receives access to a system that is similar in every way except the addition of the possibility to share mood with team members.

### 7.7.2 Virtual Office Implications

In this chapter we presented an empirical study on microblogging with mood indicators in a distributed team. This study contributes to answering the third research question, namely:

**Research Question 3**
*What is the value of microblogging with mood-indicators in global software engineering?*

From the empirical research presented in this chapter we could derive the following requirements of a virtual office:

**Req 14. Facilitate sharing information about a new topic**
*Software engineers should be able to share information about a new topic to stay current on each other's experiences and latest exploits.*

**Req 15. Facilitate responding to an existing topic**
*Software engineers should be able to respond to an existing topic to add information or to answer a question.*

**Req 16. Facilitate sharing mood**
*Software engineers should be able to express their mood, for example by specifying a happiness score ranging from totally unhappy to totally happy.*

# Part IV

# Information Needs in a Virtual Office

Chapter 8

# When to Interrupt Global Software Engineers to Provide them with What Information

*Software engineering is a highly collaborative activity in which knowledge about the work context is essential to collaborate effectively. Acquiring such knowledge is difficult in a distributed setting, since software engineers have to manually analyze, filter and combine available information in order to acquire a sufficient level of awareness. Therefore, it seems beneficial to construct a mechanism which automatically regulates information based on both the current activity of a software engineer and the importance of the new information. In this chapter we present an Estimate-Talk-Estimate study, with experienced software engineers, in which we studied both (i) what information software engineers want to know immediately and (ii) when software engineers do not mind to be interrupted with such information. The main findings include a list of information items which software engineers want to be immediately informed about, and a list of activities during which software engineers prefer not to be interrupted.*

This chapter is strongly based on our publication *"When to Interrupt Global Software Engineers to Provide them with What Information"* in the Proceedings of the 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2013). This publication is co-authored by Rini van Solingen.

## 8.1 Introduction

Software engineers need information about the context in which they are working to be able to collaborate effectively with their colleagues [Sch02, Syr97]. In the traditional co-located setting all information is available in a single place, the office building, and is exchanged relatively passively and unobtrusively between all employees present at that location [Sch02, Gam12, Fog05]. In a distributed setting, however, this information can only be exchanged by using technological support. The (global) software engineering community has developed many technological solutions to support globally dispersed teams in performing their tasks [PR12, Sar10]. However, most of these solutions only support a specific type of information and this information cannot be processed by other solutions directly [PR12]. Therefore, global software engineers have to manually analyze, filter and combine available information to acquire a sufficient level of awareness.

To spread awareness relatively passively and unobtrusively in a distributed setting as well, this analytical process of accessing, combining and filtering available information needs to be automated, see chapter 2. In essence we need a mechanism which regulates information based on the context it encloses: *'Virtual Office Walls'*. Such virtual office walls have the potential to filter the information, noises and distractions software engineers face in a co-located setting. These auto-erecting walls are unfeasible in real life, but can be created in tooling for (global) software engineers, since such tooling has access to information on what everyone is working on. In this chapter we focus on what information software engineers want to know immediately and during which activities they prefer not to be interrupted. As such the goal of this chapter is:

*"To find out how to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer"*.

This chapter is structured as follows. First in section 8.2 we discuss background information and related work of this research. Following this, in section 8.3 we discuss the method of data collection we used in this study. Subsequently, we present the findings and results in section 8.4 and 8.5. In section 8.6, we reflect on these findings and discuss the most important results. Next, in section 8.7, we discuss the threats to the validity of this study. Finally, we present the conclusions of this research and discuss future research in section 8.8.

## 8.2 Background and Related Work

Software engineering is a highly collaborative activity in which knowledge about the context in which you are working is essential to properly collaborate with others [Sch02, Syr97]. Having access to such knowledge, in literature referred to as *'awareness'* [Dou92], is essential since software engineers need to coordinate their effort to produce a functional system. In the traditional co-located setting this information is exchanged relatively passively and unobtrusively [Sch02, Fog05], so much of this information is naturally propagated to all the members of the team. In a distributed setting, however, sharing such information becomes unfeasible without technological support [Dul10]. Therefore, the (global) software engineering community has developed many solutions to provide globally dispersed teams with all the information they might need. Both [PR12] and [Lan10] provide an overview of some of the tools used for global software engineering. These developments have led to one of the main challenges in the context of GSE, the lack of integration [PR12].

When all information is integrated into a single solution, software engineers need to abstract useful information without experiencing an overload of information [Sim96]. Tell and Babar [Tel12] propose to use the Activity Theory to both structure and describe activities in software engineering processes. This makes it possible to determine what information is relevant when performing a specific collaborative task. However, this does not imply software engineers should be immediately informed of this information. Because interrupting software engineers during their work can significantly reduce a software engineer's efficiency [Sol98]. It is also interesting to understand the factors contributing to self-interuption in open office environments [Dab11, Mar08]. In his research Fischer derived a multidimensional framework that serves to identify research challenges, guidelines and design trade-offs for systems supporting awareness [Fis12]. In this chapter we focus on what information software engineers want to know immediately and when they prefer not to be interrupted.

## 8.3 Method

The goal of this study is: *"To find out how to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer"*. To be able to reach this goal we identified the following research questions:

RQ 1. *"What information do software engineers want to know immediately?"*

RQ 2. *"During which activities do software engineers prefer to be interrupted to provide them with information?"*

The outcomes of these two research questions could be contradicting. Since, on the one hand, it is likely participants indicate that they are interested in direct updates of information. On the other hand, it is also likely that they prefer not be interrupted at that specific time. Therefore we are also interested in the following research question:

RQ 3. *"What information do software engineers want to know immediately, even though they are performing an activity during which they prefer not to be interrupted?"*

Figure 8.1 provides a visual representation of these three research questions, in which their mutual relationship is shown.

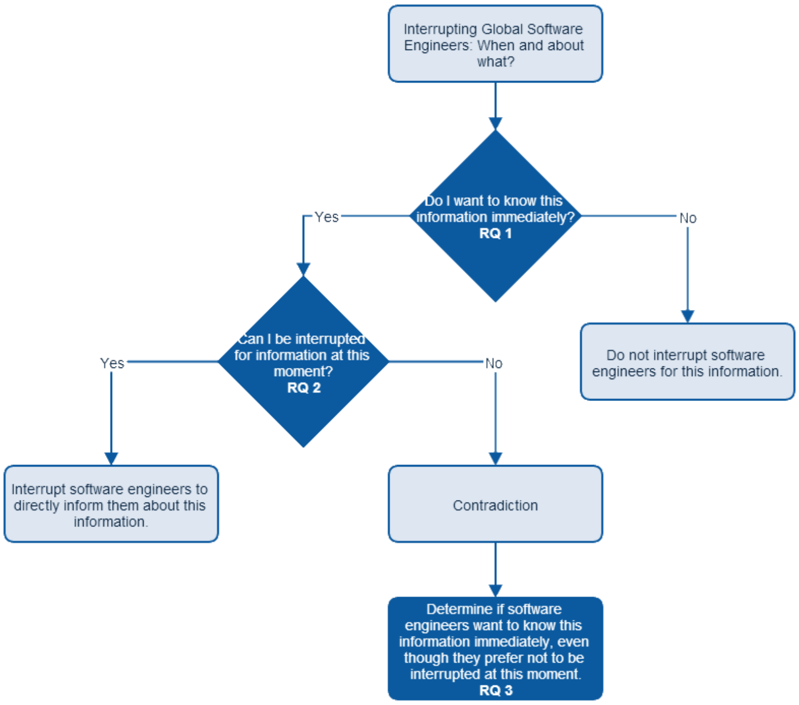

Interrupting Global Software Engineers: When and about what?
Do I want to know this information immediately? RQ 1
Yes
No
Can I be interrupted for information at this moment? RQ 2
Yes
No
Do not interrupt software engineers for this information.
Interrupt software engineers to directly inform them about this information.
Contradiction
Determine if software engineers want to know this information immediately, even though they prefer not to be interrupted at this moment. RQ 3

**Figure 8.1:** *Mutual relationship of the research questions*

These research questions span a wide range of domains, since we are interested in (i) information items useful to software engineers, (ii) common activities of software engineers, (iii) the distinction between information of which software engineers do want or do not want to be immediately informed, and finally (iv) the distinction between activities during which software engineers prefer to be or prefer not to be interrupted. Because the focus of this research is on differentiating between what information software engineers do want or do not want to know immediately and differentiating between activities during which software engineers prefer to be or prefer not to be interrupted, we decided to use the information items and activities defined in a generic life cycle model. We have looked at several of these models, including the CMMI for development, ISO 12207. IEEE 1074 and MIL 498, and could not identify significant differences considering the goal of this study. We chose to use the CMMI for development model[Tea10], because its practice oriented design and because we expect the used terminology best matches with the industrial experts. In the CMMI for development five process areas are defined: Requirements Development, Technical Solution, Verification, Validation and Product Integration. These process areas consist of multiple specific goals which all describe a unique characteristic that must be present to satisfy this area. These specific goals in turn, consist of multiple practices which are important activities to achieve the associated goal.

Now we have a structured list of information items and activities, we have to find out what information software engineers want to know immediately and during which activities they do not mind to be interrupted. Therefore, we use a structured communication technique which allows study participants to systematically deal with these issues. We use the Estimate-Talk-Estimate method [Gus73]. The main reason to use this method is that decisions made by a structured group of individuals are more accurate than individual judgments [Gus73]. Another reason to use this method is that a combination of nominal and interacting group processes is desirable in judgmental problem solving [Vro69, Dun64, Gus73]. In the remainder of this section we describe the criteria used to select study participants and the process we used to gather the empirical data.

### 8.3.1 Members of the Group of Participants

Participants in a Estimate-Talk-Estimate study are asked to provide reasons for their decisions and to respond to the decisions made by the other participants. Based on this information participants can revise their opinions. Therefore, it is essential the members of this expert study have different backgrounds so they can provide each other with new information[Gre07]. This makes choosing the appropriate subjects an important step in the entire process because it directly relates to the quality of the results generated [Oko04].

We used the following criteria to select the members of the group of participants:

1. At least five years of experience in global software engineering
2. Currently working as a software engineer (e.g. architect, tester, designer or programmer)
3. Currently working at least half of the time on software engineering tasks
4. Masters the English language
5. Ability to conceptually argue about global software engineering

Selecting study participants based on one or more characteristics is called purposive sampling [Bab12]. We initially sent out an invitation to 37 software engineers who met these criteria. We also asked them to forward the invite to other software engineers that meet this profile, however we only allowed up to two participants of the same company. Finally, ten participants were willing to participate in this study, see table 8.1. This number of participants corresponds to the number of experts recommended for a Delphi panel [Oko04].

| **Expert** | **Role** | **Professional background** |
|---|---|---|
| 1 | Technical Lead | Working in a large team of about 50 people spanning locations in the Netherlands, India, Scotland and Ukraine |
| 2 | Engineer | Working in a small distributed team of 3 people |
| 3 | Engineer | Working in a small distributed team of 6 people mainly working from home. |
| 4 | Technical Lead | Working at a large department with over 200 people, spanning multiple locations: The Netherlands, Belgium, USA, Ukraine and Malaysia |
| 5 | Project Manager | Working in a large team of about 60 people in Amsterdam, working together with departments in New York, San Jose, and Kiev |
| 6 | Engineer | Working in a small distributed spanning two locations: Delft and Kiev |
| 7 | Architect | Working in a large team of about 50 people spanning four locations: Delft, Moscow, Houston and Albuquerque |
| 8 | Engineer | Working in a small distributed teams ranging from 3 to 6 people |
| 9 | Engineer | Working in a small distributed team of 7 people working from home 1 or 2 days a week |
| 10 | Architect | Working in a small distributed team of 6 people mainly working from home |

**Table 8.1:** *Members of the group of participants*

### 8.3.2 Estimate-Talk-Estimate Approach

An Estimate-Talk-Estimate study [Gus73] has much in common with a regular Delphi study [Dal63]. Both studies rely on a panel of experts who answer multiple questions in two or more rounds. After each round the forecasts of the experts and the reasoning behind their judgments are discussed to encourage the experts to revise their earlier answers based on the argumentation of others. The goal of both studies is to reach consensus on a predefined list of items. The main difference between these two studies is that in a Delphi study a facilitator provides an anonymous summary of the expert's judgments after which the experts can revise their judgments anonymously, while in the Estimate-Talk-Estimate study all judgments are visible to everyone and the experts discuss their judgments with each other. We have chosen to use an Estimate-Talk-Estimate approach since it is expected that interactive group processes contribute to a higher quality of the estimates. In their research Gustafson et al. [Gus73] emphasize written feedback appears to lead to a reduction in the quality of estimates.

The Estimate-Talk-Estimate study we conducted consists of multiple rounds to reach consensus, see figure 8.2. During *'Round 1'*, the members of the expert group remotely completed a questionnaire, see Appendix G. The questionnaire consists of two parts. In the first half, participants were asked to order the practices from most important to least important and distinguish between practices of which they want to be or not want to be immediately informed. In the second half the participants were asked to order the practices based on how disturbing an interruption would be and distinguish between practices during which they prefer to be or prefer not to be interrupted. After this round classifications of practices were accepted if at least 80% of the experts agreed with each other. So, even if a majority of participants, either six out of ten or seven out of ten, agreed with each other we did not accept that classification as strong enough.

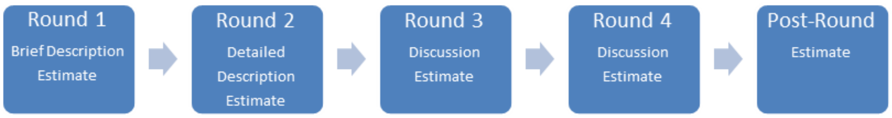


**Figure 8.2:** *Rounds of the study*

The resulting practices, on which no consensus was reached, were considered in *'Round 2'*. *'Round 2'*,*'Round 3'* and *'Round 4'* consist of two parts and were conducted during a two-hour meeting at the Delft University of Technology in the Netherlands. During this meeting both authors were present. One of the authors took the role of moderator while the other took notes. The second round started with a detailed description of the practice at hand, to clarify its meaning. Next, the experts were asked to classify this practice. To reduce the risk of influencing the other participants, we used the planning poker procedure [Gre02]. Each of the experts lays a card face down, a zero (*'no'*) or an one (*'yes'*), representing their estimate. Next, they all simultaneously call out their card by turning them over. Again classifications are accepted if at least 80% of the experts agreed with each other. *'Round 3'* and *'Round 4'* started with a time-boxed discussion in which both arguments in favor of and arguments against being informed immediately or being interrupted were discussed. During these discussions, the moderator had to ensure that everyone participated and had a chance to speak [Gib97]. To stimulate an evenly contribution of the participants, we first provided the minority of the group the opportunity to explain their judgments, after which the majority of the group had the chance to explain their judgments *'Round 3'*. In *'Round 4'* first the majority of the group was given the opportunity to convince the others, after which the minority of the group had the chance to respond. Both rounds were concluded by a re-estimation of the current practice. Table 8.2 outlines, the total number of practices, the number of practices on which consensus was reached, and the number of practices on which no consensus was reached for each of the four rounds.

| | *Round 1* | *Round 2* | *Round 3* | *Round 4* |
|---|---|---|---|---|
| Number of practices | 80 | 53 | 22 | 13 |
| Accepted practices | 27 | 31 | 9 | 3 |
| Undecided practices | 53 | 22 | 13 | 10 |

**Table 8.2:** *Results of the Estimate-Talk-Estimate study*

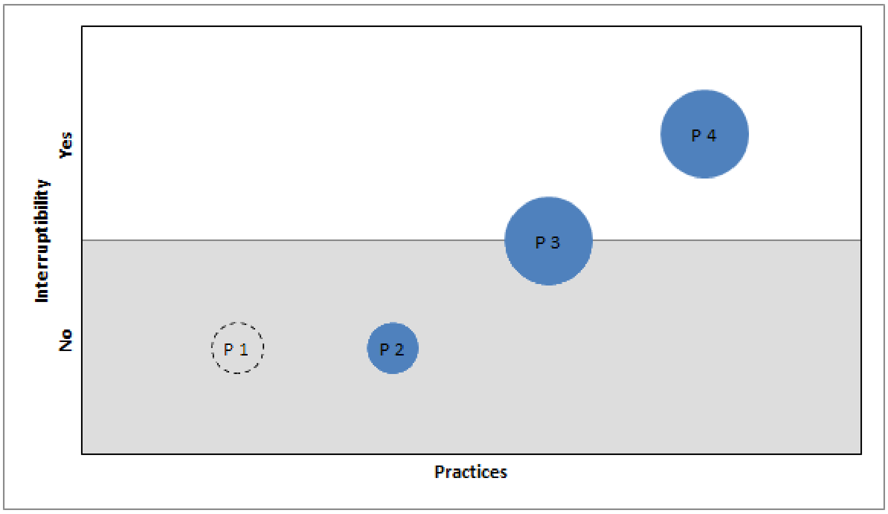
Finally, in the *'Post-Round'* the study participants remotely completed a questionnaire, see Appendix H. In this questionnaire we asked some general questions, process related questions and research related questions about the introduced contradictions. We asked the participants to indicate both (i) what information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted, and (ii) whether or not they prefer to be interrupted with information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted. The participants already reached consensus on six of the twelve practices after this initial round (*'Post-Round'*).

## 8.4 Findings Estimate-Talk-Estimate Study

To structure the findings of the Estimate-Talk-Estimate study, we use the engineering process areas defined in the CMMI, namely, Requirements Development, Technical Solution, Verification, Validation and Product Integration [Tea10]. For each of these process areas we briefly describe its purpose, goals and practices. Subsequently, we discuss for each of the practices (i) whether or not software engineers want to be informed of such information immediately and (ii) whether or not software engineers prefer to be interrupted while performing activities corresponding to the practice at hand. The results of these classifications are presented in a uniform fashion to provide the reader with a clear overview of the results. A complete overview of the classifications of all rounds can be found online at `http://aspic.nl/vow/classifications.pdf`. An illustrative example of such a representation is shown in figure 8.3. This figure shows some illustrative classifications we have made for some of the activities you perform in the morning before carpooling to work with a colleague. This representation consists of two parts. The first part is a table summarizing both the results of what information about your colleague you want to know immediately, and the results of the classifications of activities during which you do not mind to be interrupted. In the second column the results of what information about your colleague you want to know immediately is shown. In this column a *'✓-sign'* indicates that the participants agreed they want to be informed immediately of information regarding

| **Practice** | Inf | Int |
|---|---|---|
| P 1. Waking up | ? | ✗ |
| P 2. Taking a shower | ✗ | ✗ |
| P 3. Having breakfast | ✓ | ? |
| P 4. Leaving your home to go to work | ✓ | ✓ |

**(a)** *Table containing both [Inf] Information you want to know immediately when carpooling to work, and [Int] Activities during which you do not mind to be Interrupted*



**(b)** *Visualization of the results*

**Figure 8.3:** *Illustrative classification of morning activities before carpooling to work with a colleague*

the associated practice. A *'✗-sign'*, however, indicates they do not want to be informed immediately. When the participants did not agree with each other and have not reached consensus a *'?-sign'* is used to indicate this. In the third column the results of the activities during which software engineers do not mind to be interrupted are summarized. A *'✓-sign'* indicates that they agreed they do not mind to be interrupted at that moment, a *'✗-sign'* indicates they prefer not to be interrupted and a *'?-sign'* again indicates they did not reach consensus. The second part of this representation is a figure which visualizes the results of both classifications. In this figure each circle represents a single practice. The radius of the circle is used to depict whether or not the experts want to be immediately informed of information regarding the practice. A small radius indicates that the experts do not want to be immediately informed of that practice. A large radius, however, indicates that the experts do want to be immediately informed of that practice. When the circle has a small radius, is not filled and has a dashed border, the participants did not agree with each other. The location of the circle, in turn, depicts whether or not you prefer to be interrupted. A circle in the upper area concerns a practice during which you do not mind to be interrupted. A circle in the bottom area concerns a practice during which you prefer not to be interrupted. When a circle is placed on the border between these two areas the participants did not reach consensus about that practice.

### 8.4.1 Requirements Development

The first engineering process area we discuss is *'Requirements Development'*. The purpose of this area is to elicit, analyze and establish customer, product and product component requirements [Tea10]. The practices of this area belong to one of the following three goals:

SG 1 *Develop Customer Requirements*
Stakeholder needs are translated into a set of customer requirements

SG 2 *Develop Product Requirements*
Customer requirements are translated into a set of product requirements

SG 3 *Analyze and Validate Requirements*
Analyze and validate both the customer and product requirements with respect to the end user his intended environment

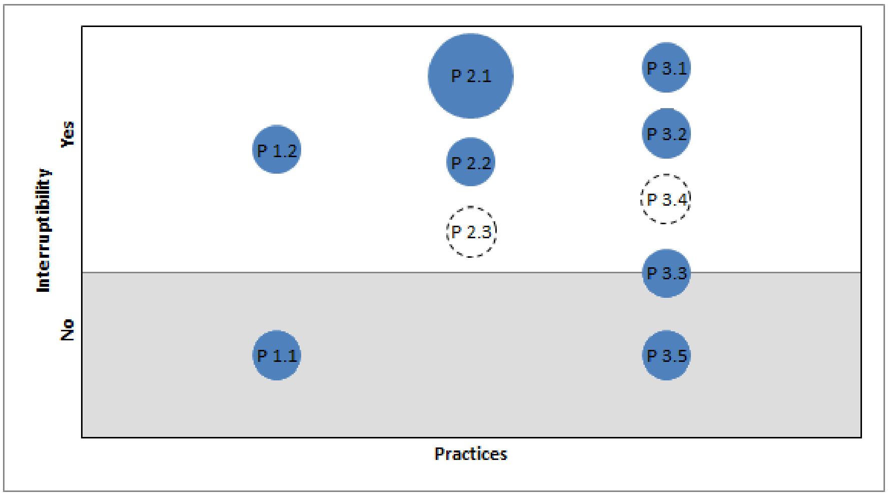
#### Information updates

Firstly, we asked the participants to order the practices based on importance, regardless of their current activity, and differentiate between the practices of which they want to be informed immediately and the practices of which they do not. The results of these classifications, after four rounds, are shown in figure 8.4a. In this table it can been seen that participants of the study indicated that they only want to be informed of new information regarding the establishment of product and product component requirements (SP 2.1). Furthermore, it can be seen that participants do not have to be informed immediately of new information regarding practice SP 1.1, SP 1.2, SP 2.2, SP 3.1, SP 3.2, SP 3.3 and SP 3.5. Finally, the participants did not reach consensus on SP 2.3 and SP 3.4.

During the face-to-face meeting a extensive discussion took place about practice SP 2.3 on which no consensus was reached. The arguments used in favor of being immediately informed about identifying interface requirements focus on the urgency of this kind of information. One of the participants said: *"I want to be informed immediately when such information becomes available"*. Arguments used against this practice focus on the fact that this is not time crucial information, which can be illustrated by the following statement: *"This kind of information can wait"*. Overall the participants indicated that whether or not they want to be informed of such information strongly depends on the impact of the specific interface.

| **Practice** | Inf | Int |
|---|---|---|
| *SG 1. Develop Customer Requirements* | | |
| SP 1.1 Elicit Needs | ✗ | ✗ |
| SP 1.2 Transform Stakeholder Needs into Customer Requirements | ✗ | ✓ |
| *SG 2. Develop Product Requirements* | | |
| SP 2.1 Establish Product and Product Component Requirements | ✓ | ✓ |
| SP 2.2 Allocate Product Component Requirements | ✗ | ✓ |
| SP 2.3 Identify Interface Requirements | ? | ✓ |
| *SG 3. Analyze and Validate Requirements* | | |
| SP 3.1 Establish Operational Concepts and Scenarios | ✗ | ✓ |
| SP 3.2 Establish a Definition of Required Functionality and Quality Attributes | ✗ | ✓ |
| SP 3.3 Analyze Requirements | ✗ | ? |
| SP 3.4 Analyze Requirements to Achieve Balance | ? | ✓ |
| SP 3.5 Validate Requirements | ✗ | ✗ |

**(a)** *Table containing both [Inf] information software engineers want to know immediately, and [Int] activities during which software engineers do not mind to be Interrupted*



**(b)** *Visualization of the results*

**Figure 8.4:** *Requirements Development*

## Interruptibility

Subsequently, we asked the participants to order the practices based on how disturbing an interruption would be while performing activities corresponding to that practice, regardless of the content of the interruption. Again we asked the participants to differentiate between practices during which they do not want to be interrupted and practices in which it is acceptable to be interrupted. Figure 8.4a provides an overview of the classifications of the practices of the requirements development area. This table shows that the participants indicated that they do not want to be interrupted while performing activities corresponding to elicit stakeholder needs and validating requirements (SP 1.1 and SP 3.5). Next, the participants did not reach consensus on practice SP 3.3, analyzing requirements, while they indicated that they do not mind to be interrupted during all other practices.

During the discussion the participants unanimous indicated that they do not want to be interrupted while they are eliciting the needs of the customer (SP 1.1). They indicated that the activities corresponding to this practice are mainly of a highly interactive nature.

### 8.4.2 Technical Solution

The second process area we discuss is the *'Technical Solution'* area. The purpose of this area is to select, design and implement solutions to the identified requirements [Tea10]. This area consists of the following three goals:

SG 1 *Select product component solutions*
Product or product component solutions are selected from alternative solutions

SG 2 *Develop the design*
Product or product component designs are developed

SG 3 *Implement the product design*
Product components are implemented from their designs

#### Information updates

The classifications of the practices necessary to achieve the goals of the technical solution area are shown in figure 8.5a. In this table it can been seen that the participants are interested in new information of four of the eight practices, namely information about selecting product component solutions (SP 1.2), designing the product (SP 2.1), designing the interfaces (SP 2.3) and implementing the design (SP 3.1).

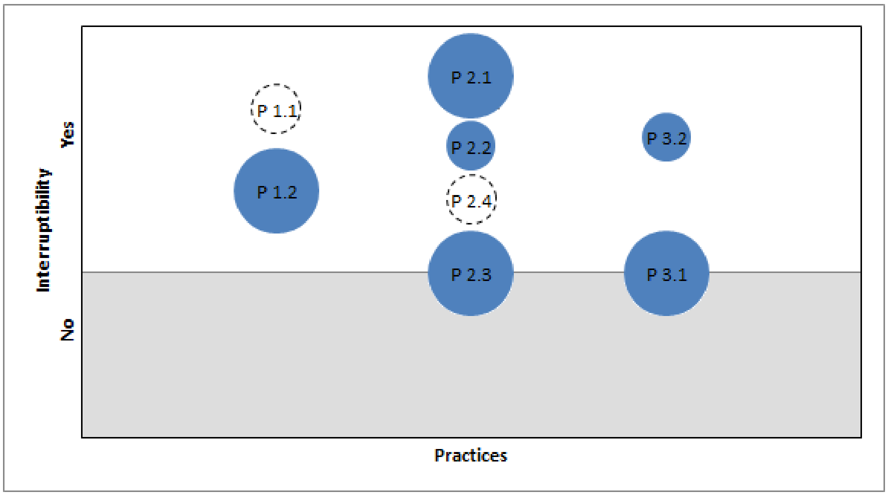
During the meeting practices SP 1.1 and SP 2.4 were most discussed. Both practices consist of activities in which multiple analyses are conducted. On the one hand several possible solutions are examined (SP 1.1) while on the other hand many make, buy and reuse analyses are performed (SP 2.4). Some of the participants argued that they *"definitely want to know such information immediately, so that they are able to influence the outcomes"* while others argued that they *"do not necessarily want to be informed of this information immediately, but are particularly interested in the choices being made"*.

#### Interruptibility

Also for this area, we asked the participants to indicate during which activities they do not mind to be interrupted. It is interesting to note that for none of the activities needed to satisfy this area, the participants indicated that they prefer not be interrupted.

| **Practice** | Inf | Int |
|---|---|---|
| *SG 1. Select Product Component Solutions* | | |
| SP 1.1 Develop Alternative Solutions and Selection Criteria | ? | ✓ |
| SP 1.2 Select Product Component Solutions | ✓ | ✓ |
| *SG 2. Develop the Design* | | |
| SP 2.1 Design the Product or Product Component | ✓ | ✓ |
| SP 2.2 Establish a Technical Data Package | ✗ | ✓ |
| SP 2.3 Design Interfaces Using Criteria | ✓ | ? |
| SP 2.4 Perform Make, Buy, or Reuse Analyses | ? | ✓ |
| *SG 3. Implement the Product Design* | | |
| SP 3.1 Implement the Design | ✓ | ? |
| SP 3.2 Develop Product Support Documentation | ✗ | ✓ |

**(a)** *Table containing both [Inf] information software engineers want to know immediately, and [Int] activities during which software engineers do not mind to be Interrupted*



**(b)** *Visualization of the results*

**Figure 8.5:** *Technical Solution*

Furthermore it is noteworthy to mention one of the arguments made during the discussion regarding whether or not you prefer to be interrupted when you are implementing the design (SP 3.1). During this discussion some participants indicated that they were *"in the zone"* when they are implementing specific functionality. They emphasize that interruptions in such a mental state have a huge impact, since it is extremely difficult to reach such a mental state again.

### 8.4.3 Verification

The third engineering process area which we discuss is the *'Verification'* area. The purpose of this area is to ensure that work products meet their specified requirements [Tea10]. This area consists of three goals:

SG 1 *Prepare for verification*
Preparation for verification to ensure that verification provisions are embedded in product requirements, designs, implementation and schedules

SG 2 *Perform peer reviews*
Peer reviews involve a methodical examination of work products to identify defects and recommend changes

SG 3 *Verify selected work products*
Verification methods, procedures and criteria are used to actually verify selected work products

#### Information updates

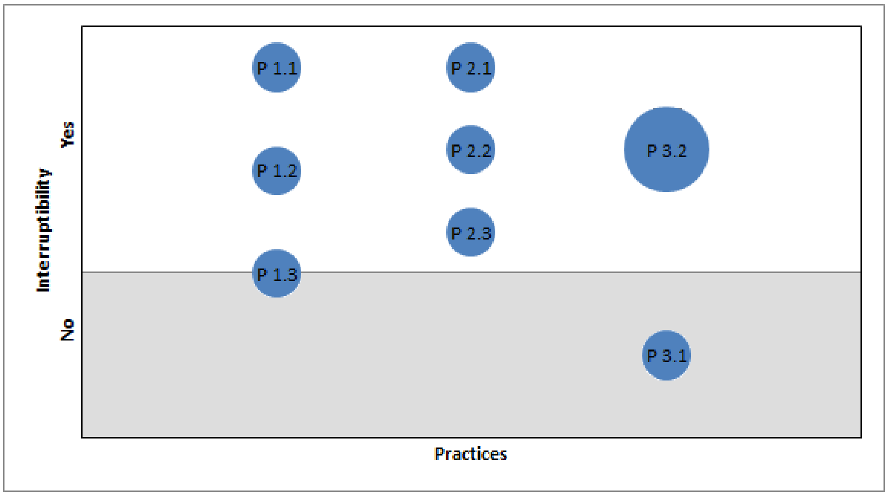
The participants of this study only indicated that they want to be immediately informed of information regarding the verification results (SP 3.2), as can been seen in figure 8.6a. There was, however, a discussion about the information regarding the analysis of peer review data (SP 2.3). One of the participants stated that such information is useful to him, especially when it concerns his own work, and that he immediately wants to be informed of such information. Another participant agreed, but said: *"I do not want to be informed of the results of the peer reviews of everyone else"*. So, overall they indicated that they are not interested in such information since they are only interested in a specific part of the information.

#### Interruptibility

The classifications of whether or not they prefer to be interrupted while performing activities related to the verification area are also depicted in figure 8.6a. The participants reached consensus on practice SP 3.1, performing the verification, for which they agreed they do not want to be interrupted. The discussion of practice SP 1.3, establish verification procedures and criteria, resulted in an interesting finding, namely that the participants use different processes to establish these procedures and criteria. Some of the participants establish these in close collaboration with the customer and therefore they would not like to be interrupted. Other participants establish these procedures and criteria on their own and therefore do not mind to be interrupted.

| **Practice** | Inf | Int |
|---|---|---|
| *SG 1. Prepare for Verification* | | |
| SP 1.1 Select Work Products for Verification | ✗ | ✓ |
| SP 1.2 Establish the Verification Environment | ✗ | ✓ |
| SP 1.3 Establish Verification Procedures and Criteria | ✗ | ? |
| *SG 2. Perform Peer Reviews* | | |
| SP 2.1 Prepare for Peer Reviews | ✗ | ✓ |
| SP 2.2 Conduct Peer Reviews | ✗ | ✓ |
| SP 2.3 Analyze Peer Review Data | ✗ | ✓ |
| *SG 3. Verify Selected Work Products* | | |
| SP 3.1 Perform Verification | ✗ | ✗ |
| SP 3.2 Analyze Verification Results | ✓ | ✓ |

**(a)** *Table containing both [Inf] information software engineers want to know immediately, and [Int] activities during which software engineers do not mind to be Interrupted*



**(b)** *Visualization of the results*

**Figure 8.6:** *Verification*

## 8.4.4 Validation

The fourth area we discuss is the *'Validation'* area. The purpose of this engineering area is to demonstrate that a product fulfills its intended use [Tea10]. This area consists of two goals:

SG 1 *Prepare for validation*
Preparation for validation include selecting products for validation and establishing and maintaining the validation environment

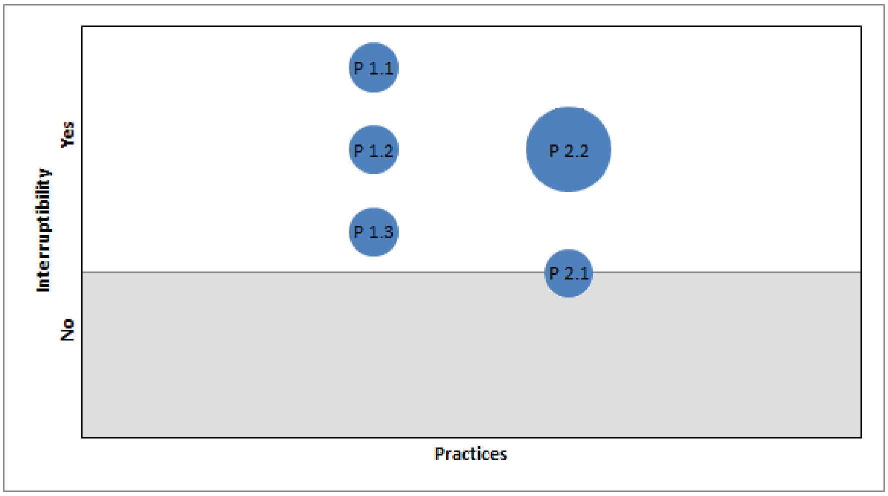
SG 2 *Validate product or product components*
Validation methods, procedures and criteria are used to actually validate selected work products

## Information updates

The classifications of the validation practices are very similar to the classification of the verification practices, see figure 8.7a. Again, the participants indicated that they do not want to be informed immediately of information regarding the preparation of the validation process (SP 1.1, SP 1.2 and SP 1.3). While they do want be notified of information about the analysis of the validation results (SP 2.2).

| **Practice** | Inf | Int |
|---|---|---|
| *SG 1. Prepare for Validation* | | |
| SP 1.1 Select Products for Validation | ✗ | ✓ |
| SP 1.2 Establish the Validation Environment | ✗ | ✓ |
| SP 1.3 Establish Validation Procedures and Criteria | ✗ | ✓ |
| *SG 2. Validate Product or Product Components* | | |
| SP 2.1 Perform Validation | ✗ | ? |
| SP 2.2 Analyze Validation Results | ✓ | ✓ |

**(a)** *Table containing both [Inf] information software engineers want to know immediately, and [Int] activities during which software engineers do not mind to be Interrupted*



**(b)** *Visualization of the results*

**Figure 8.7:** *Validation*

## Interruptibility

It is interesting to see that for none of the practices of the validation area the participants indicated that they prefer not be interrupted. The only practice on which they did not reach consensus is practice SP 2.1, performing the actual validation. Some of the participants argued that they prefer not to be interrupted by colleagues since this process is performed in close collaboration with the customer. Other participants, however, argued that they do not mind to be interrupted. Since, performing the validation can be a time-consuming activity, which can take days, it is unfeasible not to be disrupted at all.

### 8.4.5 Product Integration

Finally, we discuss the *'Product Integration'* area. The purpose of this area is to assemble the product from the product components, ensure that the product fulfills all requirements, and deliver the product [Tea10]. This area consists of the following three goals:

SG 1 *Prepare for product integration*
Preparation for product integration includes establishing an integration strategy, the integration environment and the integration procedures and criteria.

SG 2 *Ensure product interface compatibility*
Effective management of product component interfaces helps ensure that implemented interfaces will be complete and compatible

SG 3 *Assemble product components and deliver the product*
Integration of product components proceeds according to the product integration strategy and procedures.

#### Information updates

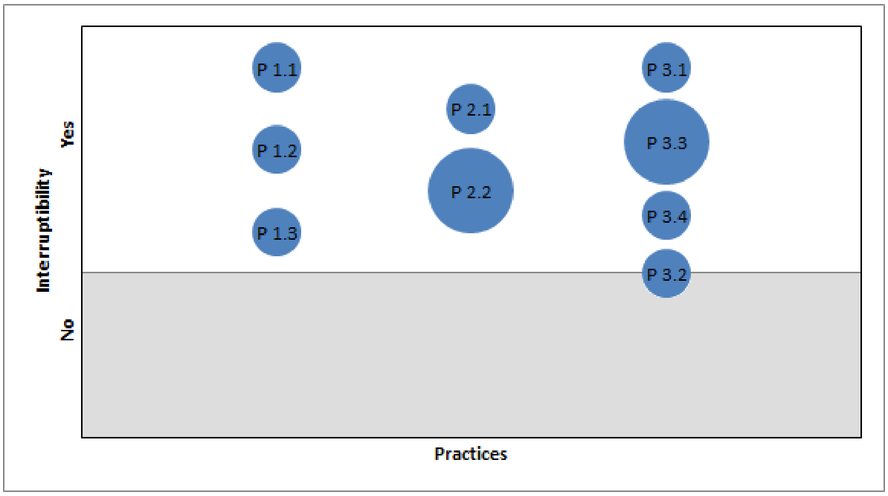
In figure 8.8a the results of the discussions are shown. In this table it can been seen that the experts are only interested in direct updates of information regarding managing interfaces (SP 2.2) and evaluation of assembled product components (SP 3.3). Again they are less interested in information about the preparation phase.

#### Interruptibility

Finally, we discuss the classifications regarding interruptibility. The participants reached consensus for all but one practice. The only practice on which they did not reach consensus is practice SP 3.2. The two main arguments in the discussion on this practice regarding assembling product components are: (i) *"assembling product components is really important and requires a high level of concentration, so I do not want to be interrupted"* and (ii) *"I do not mind to be interrupted, since assembling product components does not require specialist knowledge"*. Overall the participants concluded that whether or not you prefer to be interrupted during the assembly of product components strongly depends on personal preference.

| **Practice** | Inf | Int |
|---|---|---|
| *SG 1. Prepare for Product Integration* | | |
| SP 1.1 Establish an Integration Strategy | ✗ | ✓ |
| SP 1.2 Establish the Product Integration Environment | ✗ | ✓ |
| SP 1.3 Establish Product Integration Procedures and Criteria | ✗ | ✓ |
| *SG 2. Ensure Interface Compatibility* | | |
| SP 2.1 Review Interface Descriptions for Completeness | ✗ | ✓ |
| SP 2.2 Manage Interfaces | ✓ | ✓ |
| *SG 3. Assemble Product Components and Deliver the Product* | | |
| SP 3.1 Confirm Readiness of Product Components for Integration | ✗ | ✓ |
| SP 3.2 Assemble Product Components | ✗ | ? |
| SP 3.3 Evaluate Assembled Product Components | ✓ | ✓ |
| SP 3.4 Package and Deliver the Product or Product Component | ✗ | ✓ |

**(a)** *Table containing both [Inf] information software engineers want to know immediately, and [Int] activities during which software engineers do not mind to be Interrupted*



**(b)** *Visualization of the results*

**Figure 8.8:** *Product Integration*

## 8.5 Findings Post-Round

In this section we present the findings of the post questionnaire (A complete overview of the classifications can be found online at `http://aspic.nl/vow/post-questionnaire-classifications.pdf`. The findings related to the first two research questions are contradicting since, on the one hand, for some practices the experts indicated that they prefer not to be interrupted when performing activities corresponding to that practice, while on the other hand the experts indicated that they want to be informed of information regarding several practices immediately. In order to elaborate on the needs of software engineers, we asked each of the participants to indicate both (i) what information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted, and (ii) whether or not they prefer to be interrupted with information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted.

The participants reached consensus on five of the nine practices regarding the information they want to know immediately. For each of these practices, RD 2.1, TS 3.1, VER 3.2, PI 2.2 and PI 3.3, they agreed they do not want to be informed immediately when they are performing an activity during which they prefer not to be interrupted. Furthermore, it is interesting to note the experts have not yet reached consensus on a practice of which they do want to be immediately informed.

We also asked the experts to indicate whether or not they prefer to be interrupted with information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted. The only practice on which the experts reached consensus is practice SP 1.1 in the area of requirements development. They agreed they prefer not be interrupted while performing activities related to this practice even if it concerns information of which they want to be informed immediately.

These results are combined in table 8.3. In this table a *'✗-sign'* indicates software engineers do not want to be immediately informed of information regarding the practice depicted in the column, while they are performing activities corresponding to the practice depicted in the row. A *'?- sign'* indicates the participants have not yet reached consensus. Currently, the participants have not reached consensus on a practice of which they want to be immediately informed.

| | | RD | TS | | | | VER | VAL | PI | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 2.1 | 1.2 | 2.1 | 2.3 | 3.1 | 3.2 | 2.2 | 2.2 | 3.3 |
| RD | 1.1 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| | 3.5 | ✗ | ? | ? | ? | ✗ | ✗ | ? | ✗ | ✗ |
| VER | 3.1 | ✗ | ? | ? | ? | ✗ | ✗ | ? | ✗ | ✗ |

**Table 8.3:** *Information software engineers want to know immediately even though they prefer not to be interrupted*

## 8.6 Discussion

In this section we reflect on the findings and discuss the most important results of this study.

Firstly, we discuss what information software engineers want to know immediately, see table 8.4. From the figures in the previous section it can be seen that participants of our study only want to be informed immediately of roughly one quarter of the practices. It is interesting to notice that the experts want to be informed immediately of at least one practice of each of the five process areas. For each of these areas, they are mainly interested in direct updates of information regarding completed artifacts (e.g. requirements, design and verification results),

| **Area** | **Practice** |
|---|---|
| Requirement Development | SP 2.1 Establish Product and Product Component Requirements |
| Technical Solution | SP 1.2 Select Product Component Solutions |
| Technical Solution | SP 2.1 Design the Product or Product Component |
| Technical Solution | SP 2.3 Design Interfaces Using Criteria |
| Technical Solution | SP 3.1 Implement the Design |
| Verification | SP 3.2 Analyze Verification Results |
| Validation | SP 2.2 Analyze Validation Results |
| Product Integration | SP 2.2 Manage Interfaces |
| Product Integration | SP 3.3 Evaluate Assembled Product Components |

**Table 8.4:** *Information software engineers want to know immediately*

and are less interested in direct updates of information concerning the procedures used and environment needed to construct these artifacts. Another interesting result is the distribution of the practices, of which the participants want to be informed immediately, over the process areas. Nearly half of these practices belong to the technical solution area. Participants are not only interested in the implementation itself but are also interested in the selected solutions, the design of the component and the design of the interfaces. This is shown in figure 8.9, since we have plotted the ratio of all process areas, defined as the number of practices of which the participants want to be immediately informed, and the number of practices on which the participants reached consensus.

Secondly, we elaborate on research question 2: *"During which activities do software engineers prefer to be interrupted to provide them with information?"*. The study participants indicated in general they do not mind to be interrupted while performing software engineering related activities. They only reached consensus on three practices during which they prefer not to be interrupted, see table 8.5. These practices belong to either the requirements development area or the verification area (see figure 8.9). The main arguments used to convince others that they prefer not to be interrupted while performing a certain activity are (i) the activity is of a high interactive nature involving customers and therefore you do not want to be interrupted and (ii) performing the activity requires a high level of concentration and therefore it is difficult to again reach this mental state

| **Area** | **Practice** |
|---|---|
| Requirement Development | SP 1.1 Elicit Needs |
| Requirement Development | SP 3.5 Validate Requirements |
| Verification | SP 3.1 Perform Verification |

**Table 8.5:** *Activities during which software engineers prefer not to be interrupted*

after an interruption. Overall the participants agreed that whether or not you prefer to be interrupted during a specific activity strongly depends on the person and therefore is hard to accurately predict.

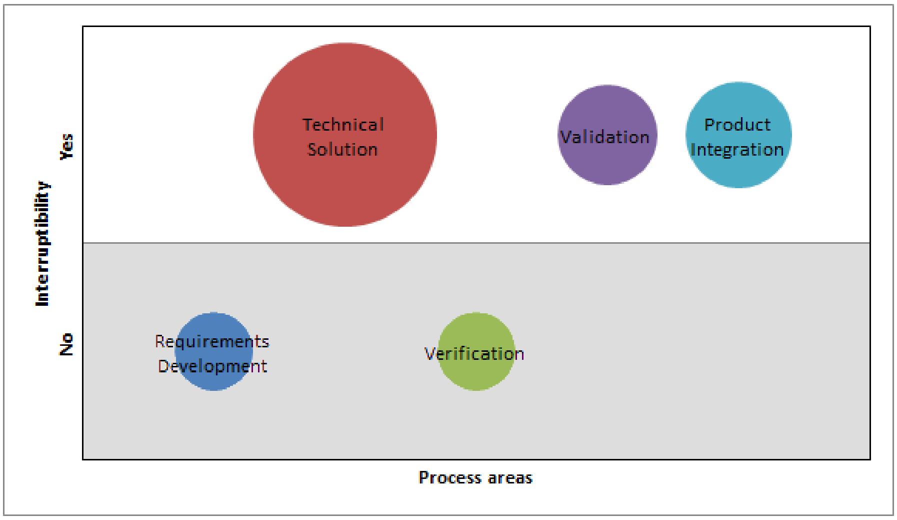


**Figure 8.9:** *Summary of the process areas*

Thirdly, research question 3 elaborates on the needs of software engineers when they are performing activities during which they prefer not to be interrupted in general. We asked the participants to indicate both (i) what information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted, and (ii) whether or not they prefer to be interrupted with information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted. The initial results of these questions indicate that software engineers do not want to be informed immediately when they are performing an activity during which they do not want to be interrupted.

Finally, we reflect on the Estimate-Talk-Estimate study we conducted, because such a study is rarely conducted in the field of software engineering. In such a study it is important that all participants have a thorough understanding of the issues being discussed to make reliable decisions. In the first round we conducted a questionnaire in which we asked the participants to classify the practices based on a brief description. In the second round we asked the participants to classify the practices on which no consensus was reached during the first round based on a detailed description. In round three and four an interactive discussion took place among the experts, after which the practices on which no consensus was reached were classified. In the post round we asked for their understanding of the practices of the five process areas. Participants indicated to have a good or very good understanding of the practices out of the following options: *'Very-Good'*, *'Good'*, *'Acceptable'*, *'Poor'* and *'Very-Poor'*. As such, all participants were able to make well informed decisions. Another important element of this study is the interactive group discussion. In these discussions participants were asked to provide reasons for their decisions and to respond to the decisions made by others, after which participants could revise their opinion. To see if this process actually took place we asked the participants if they revised their opinion based on the arguments of others. They were given five options: *'Never'*, *'Rarely'*, *'Sometimes'*, *'Very Often'* and *'Always'*. Nine of the participants indicated this was sometimes the case. Only one of the participants indicated he often changed his opinion based on the arguments of others. Based on these experiences we consider an Estimate-Talk-Estimate study is a suitable way to allow participants to systematically classify multiple issues.

## 8.7 Threats to validity

In this section we discuss the threats to validity for this study on three aspects: reliability, internal validity and external validity. We mitigated threats to reliability by providing a detailed description of the methods we used. We have described in detail how the study participants were selected and of which rounds this study consists. We also made the design of both questionnaires and the detailed descriptions of the practices available. Next to this we also make all classifications of the practices available in a anonymized form. We do this to make both our data gathering methods and the analysis of our data repeatable, and as such increase the reliability of this research. We further mitigated threats to reliability by pretesting both questionnaires with two software engineers, who did not participate in this study.

Next, there exist threats to internal validity. During this study several practices were classified during a collaborative session. These practices are relatively abstract because they could reference multiple information items and activities. To determine to which extent this influenced the results we asked the participants about their understanding of the practices. They all indicated they had a good understanding of the practices of the five process areas, one participant even stated to have a very good understanding. Since applied social research is a human activity, it is also possible social pressure influenced the outcomes of this research. Since the estimates made by the participants were not anonymous, as in a Delphi study, it is possible participants changed their behavior to fulfill the expectations of others as a result of real or imagined group pressure. Therefore, in the post questionnaire, we asked the participants if they changed their opinion because of the peer pressure of the group of experts. They were given five options: *'Never'*, *'Rarely'*, *'Sometimes'*, *'Very Often'* and *'Always'*. Seven of the participants indicated they never changed their opinion while the other three indicated this was rarely the case.

Finally, we discuss threats to external validity. External validity is of interest in studies that draw generalized conclusions. In this study we consulted ten experienced Dutch software engineers from nine different companies. To improve the external validity of the findings, further studies are needed to diversify the group under study, including for example non-Dutch participants.

## 8.8 Concluding Remarks

### 8.8.1 Conclusions

In this chapter we have reported on the empirical study we conducted on how to regulate information available to software engineers based on both the importance of the information and the current activity of the engineer. To structure this research we used the five engineering process areas defined in the CMMI [Tea10]: Requirements Development, Technical Solution, Verification, Validation and Product Integration. These areas consist of multiple goals and practices, which we used to determine (i) of what information software engineers want to be immediately informed, and (ii) during which activities software engineers do not mind to be interrupted. The outcomes of the two research directions were contradicting in some cases, since participants of this study indicated that they would like to be informed of several practices immediately while for other practices they indicated that they prefer not to be interrupted. Therefore we also researched (iii) of what information software engineers want to be immediately informed, even though they are performing an activity during which they prefer not to be interrupted.

The main contributions of this chapter are the answers to the research questions. First, we showed a list of information items of which software engineers want to be informed immediately. Subsequently, we presented a list of software engineering activities during which software engineers prefer not to be interrupted. Finally, we discussed a look-up table which can be used to determine whether or not software engineers want to be immediately informed, even though they are performing an activity during which they prefer not to be interrupted.

Next, when abstracting the findings, we can conclude that:

- Software engineers want to be immediately informed of a wide variety of information
- Software engineers are mainly interested in direct updates of information about completed artifacts
- Software engineers are especially interested in information regarding the technological solution itself
- In most cases software engineers do not mind to be interrupted to provide them with information

- Software engineers prefer not to be interrupted when they are performing activities of a highly interactive nature
- Software engineers prefer not to be interrupted when they are performing activities which require a high level of concentration
- Software engineers do not want to be immediately informed of any information when they are performing an activity during which they prefer not to be interrupted

Next steps of this research include (i) reaching consensus on information items on which no consensus was reached, (ii) reaching consensus on information items of which software engineers want to be immediately informed when performing activities during which they prefer not to be interrupted, and (iii) research the different practices in more detail, e.g. what artifacts and what specific information is needed. These results can then be used to construct virtual office walls which automatically regulate information available to a software engineer, based on both the current activity of the engineer, and the information engineers want to know immediately. Until we all work in a *'Virtual Office'* in which information is regulated by *'auto-erecting virtual office walls'*, we should carefully consider whether to interrupt a colleague to provide him or her with information. So, when in doubt, do not disturb!

### 8.8.2 Virtual Office Implications

In this empirical study we researched what information global software engineers want to know immediately, and when they prefer not to be interrupted. This study answers the fourth research question of this dissertation:

**Research Question 4**
*How to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer?*

From this empirical study we could derive the following requirements of a virtual office:

**Req 17. Facilitate informing software engineers immediately of information about completed artifacts**
*Software engineers are especially interested in immediate updates of information about completed artifacts, e.g. requirements, design, and verification results.*

**Req 18. Facilitate informing software engineers immediately of information about the technological solution**

*Software engineers are especially interested in immediate updates of information about the technological solution itself, e.g. the selected solutions, the design of the components, and the design of the interfaces.*

**Req 19. Facilitate interrupting software engineers to provide them with new information**
*Software engineers should be interrupted to immediately provide them with information they want to know immediately.*

**Req 20. Facilitate controlling the moments during which one prefers not to be interrupted**
*Software engineers should be able to control whether they prefer to be or prefer not to be interrupted, especially during activities of a high interactive nature and activities which require a high level of concentration.*

Chapter 9

# Evaluating the Impact of Virtual Office Walls

*A Virtual office wall is a mechanism which automatically regulates information to support distributed software engineers. These walls reduce the available information to only that information which is currently relevant. In this chapter we present a controlled experiment with experienced software engineers as study participants. In this experiment we study whether there is a relation between the presence of virtual office walls and the actual and perceived speed and accuracy of the work carried out by the participants. Additionally, we measured the extent in which the participants experience the presence of virtual office walls as useful. In the experiment we cannot use real software engineering work, because this would make the experiment too long to be feasible. Therefore, we decided to use fictional map tasks, like adding an arrow between two objects, which mimic important aspects of global software engineering tasks. These fictional map tasks also minimize mistakes due to differences in software programming abilities of the participants. The main findings include that virtual office walls appear to contribute to an improved awareness of co-worker synchronicity, an easier insight in what to do and a more concise overview of the work performed. These improvements mostly benefit the speed of coordination and the perception regarding overall performance.*

This chapter is based on our publication *"Auto-Erecting Virtual Office Walls a Controlled Experiment"* in the Proceedings of the 8th International Conference on Global Software Engineering (ICGSE 2013). This publication is co-authored by Rini van Solingen and Kevin Dullemond.

## 9.1 Introduction

In a co-located setting, software engineers are confronted with all kinds of information, noises, distractions, etc. that are not relevant to their work at hand. These distractions can be so severe a complete workday becomes ineffective [Sol98]. Dislocated software engineers have the advantage that tooling has the potential to filter the information delivered to them. Such tooling has access to information on what everyone is working on and can as such filter non-relevant information. Filtering information in such a fashion resembles the creation of *'moving'* office walls in a co-located fashion that move around all the time, depending on the work an engineer is carrying out. Such *'auto-erecting virtual office walls'* are unfeasible in real life but can be created in tooling for (globally) distributed software engineers. As such, these walls provide dislocated software engineers with the awareness level of a *'virtual office'*, undisturbed by information not relevant to their current activity.

The main contribution of this chapter is the indication that virtual office walls are valuable to (dislocated) software engineers. In practice this comes down to the actual speed and accuracy of their work, as well as perceptions of speed, accuracy and usefulness. Furthermore, an important contribution of this work is the construction of a controlled experimental design with tasks, resources and materials that can be used for replicating and scaling of the experiment.

Research on virtual office walls is important because it enables dislocated software engineers to better focus on the actual work and prevents distractions and manually searching for required information, see chapter 2. Such research, however, also brings an additional dimension to the research on distributed software engineering. Tooling for distributed software engineering is largely set up with the motivation to compensate for the negative consequences of distance [Car01]. However, such tools also provide opportunities to further build upon. A real-life physical wall is not able to be moved easily and has no understanding of the work an office occupant is working on. Tools for dislocated software engineers do possess such knowledge and can, as such, leverage it. Because of the promise of advantages over the co-located setting, additional benefits and added value can be delivered by removing the physical boundaries of work offices.

This chapter is structured as follows. First in section 9.2 we discuss related work regarding virtual office walls. Following this, in section 9.3 we discuss the research questions and hypotheses of the experiment. Subsequently, in section 9.4, we present the design of the experiment and discuss the variables and tooling used in this experiment. In section 9.5 we present the findings and evaluate the hypotheses based on these findings. Next, in section 9.6, we discuss the threats to the validity of this study. Finally, we present the conclusions of this research and discuss future research in section 9.7.

## 9.2 Related Work

Software engineering is a highly collaborative activity in which knowledge about the context in which you are working is essential to properly collaborate with others [Sch02, Syr97]. In literature this knowledge is commonly referred to as *'awareness'* [Dou92, Sch02]. Examples of such information items are: information about the other members of the project team, their activities and information about the progress of the project. It is essential to have a sufficient level of awareness, since software engineers need to coordinate their efforts to be able to produce a functional system.

In the traditional co-located setting this information is exchanged relatively passively and unobtrusively [Sch02, Fog05], so engineers are continuously aware of information related to their current activity. In chapter 2 we discussed this is probably caused by the design of the office building [All07], since in general an office building consists of several rooms each with its own characteristics. By moving around in the building software engineers are able to select a room with characteristics that match their needs, and as such are able to change the context of their activities.

But nowadays, both due to the globalization of business [Car99, Her01, Her07] and because people are starting to work from home more and more [Die09], people no longer share a physical work environment. In such a distributed setting exchanging information without technological support becomes unfeasible [Dul10]. So, in order to retrieve information related to their current activity software engineers need to use technological solutions. To fulfill this need, the (global) software engineering community has developed and studied a wide variety of tools, for example: Instant Messaging solutions, issue management systems and configuration management systems [PR12, Sar10].

Most of these tools, however, only support a single aspect of the development process and as a consequence many diverse tools are needed to provide software engineers with the information they need [PR12]. When they have gathered all relevant information, this information needs to be analyzed, combined and filtered manually by each software engineer to acquire the information necessary to create a context for his current activity. This process can be quite time-consuming and may result in misunderstandings, inconsistencies, incompatibilities and duplicated information [PR12]. The concept of filtering information available to software engineers to help them focus on their current task has been studied extensively. In the seventies Parnas discussed how to break up a single program into multiple independent parts to be able to develop each of them in isolation of the implementation of the other [Par76]. Other research [Epp04, Hil85] focuses on what information is necessary to software engineers and how to provide them with this information without causing an information overload.

In chapter 2 we discuss how best to provide distributed software engineers with the awareness they need. In this chapter we conclude that the analytical process of accessing, combining and filtering information needs to be automated to be able to acquire awareness in a relatively passive and unobtrusive fashion. So, in essence we need a mechanism which automatically regulates information based on the current context of a software engineer: a *'virtual office wall'*.

## 9.3 Research Questions and Hypotheses

The goal of this study is: *"to find out how valuable virtual office walls are for real-life distributed software engineers during their day-to-day activities"*. To determine this value, we measure the extent in which this kind of technical support impacts the speed and accuracy of the work, the extent in which it impacts the perception of speed and accuracy of the work, and the extent in which experienced distributed software engineers consider this kind of support useful.

To reach this goal we have formulated the following research questions for this experiment:

**RQ1** How do virtual office walls influence the speed of work?

**RQ2** How do virtual office walls influence the perception of the speed of work?

**RQ3** How do virtual office walls influence the accuracy of the work carried out?

**RQ4** How do virtual office walls influence the perception of the accuracy of the work carried out?

**RQ5** How useful is the introduction of virtual office walls in a (distributed) software engineering project?

**RQ6** Do virtual office walls make it easier to understand what is going on in a software engineering project?

In order to answer these research questions we have chosen to perform a controlled experiment with experienced distributed software engineers as study participants. The reasons we chose to conduct a controlled experiment are:

- We intended to find evidence that the introduction of virtual office walls is valuable. Therefore we need to be able to measure the influence of this introduction in isolation.
- We intended to find evidence that the actual and perceived performance differs when the amount of information is limited (as already indicated by Solingen and Valkema [Sol10] and based on the research of Prickladnicki [Pri09])

Considering these two intentions a controlled experiment with real-life software engineers with distributed experience is the best approach. Before undertaking the experiment we formulated the following hypotheses regarding the above six research questions:

**H1** The introduction of virtual office walls has a positive impact on the speed of work carried out

**H2** The introduction of virtual office walls has a positive impact on the perception of the speed of work

**H3** The introduction of virtual office walls has no impact on the accuracy of the work carried out

**H4** The introduction of virtual office walls has a strong positive impact on the perception of accuracy of the work

**H5** Software engineers consider the introduction of virtual office walls a useful feature

**H6** The introduction of virtual office walls makes it easier to differentiate between information that is relevant to the current activity of an engineer and information that is not.

Hypotheses 1 and 2 express we expect the speed of work to go up, because the reduced amount of information makes it easier to find out what to do. Hypothesis 3 expresses that we expect the accuracy of the work will not be different as this is largely determined by the specific task itself and the specific skills of each individual engineer. Hypothesis 4, however, expresses that the perception of accuracy will be influenced because the more difficult it is to see what is happening the larger the probability one might feel things are not going well. Hypotheses 5 and 6 express that we expect providing distributed software engineers with the information they need is considered useful and makes it easier to understand the current status of the project.

## 9.4 Controlled Experiment

We conduct a controlled experiment to study whether there is a relation between the presence of virtual office walls and the actual and perceived speed and accuracy of the work carried out by distributed software engineers. Additionally we measured the extent in which the participants experience the presence of virtual office walls as useful. In a controlled experiment the results obtained from two samples are compared; the results obtained from a test group and the results obtained from a control group. These two groups should be as similar as possible except for the one aspect of which the effect is being tested [Woh00]. In this section we discuss the design of the experiment, the dependent, independent and control variables, the tooling environment used, and the context in which the experiment is conducted.

### 9.4.1 Design

In this controlled experiment we examine the impact virtual office walls have on actual and perceived speed and accuracy. Additionally, we measured the extent in which participants experience the presence of virtual office walls as useful. Therefore we split the total group of study participants into two subgroups; a test group which has access to an environment in which the concept of virtual office walls is implemented and a control group which has access to an environment in which this concept is disabled. Such an experiment is referred to as a *'One factor with two treatments experiment'* [Woh00]. The distribution of participants into these groups is random but balanced based on seniority of the participants.

During the experiment both groups have to successfully complete six projects. Each of these projects consists of: (i) three randomly assigned project members (participants), made anonymous by changing their names. Each participant is only allowed to work on the projects he or she is assigned to, (ii) twelve project specific tasks, each task has a status: open, in progress or resolved, a description and a corresponding resource, and (iii) two project specific resources[1], named after a city or country. Each resource has a status; locked or unlocked, and the location of the resource.

Participants of the experiment should use the following process to successfully complete each of the six projects (see figure 9.1 for an overview):

1. *Selecting a Task:* A participant should select an open task of one of the projects he or she is assigned to. Subsequently, the engineer should verify that all tasks on which this task depends are resolved.

2. *Selecting a Resource:* When the participant has selected a task, he or she has to verify the status of the corresponding resource. If the status of the resource is unlocked the participant, can lock the resource. However, when the status of the resource is locked, the participant should select another task.

3. *Locking a Resource:* Before a participant is able to work on a task he or she should first lock the corresponding resource. If the status of the resource is locked, the participant should select another task.

4. *Assigning to a Task:* When a participant has locked the resource corresponding to the selected task he or she should assign this task to himself or herself.

5. *Working on a Task:* When a participant has both assigned himself or herself to a task and locked the resource corresponding to this task, he or she is able to start working on the task. The participant first needs to download the resource from a central repository, subsequently, the participant can

[1]Despite what is common in practice, we consider the term resource to be limited to the materials necessary to carry out the tasks

perform the task instruction, and finally the participant needs to save and upload the file to the central repository.

6. *Resolving a Task:* When a participant completed the task on which her or she was working, the participant should update the status of the task to resolved.
7. *Unlocking a Resource:* When a participant resolved the task to which he or she assigned himself or herself, the participant should also update the status of the corresponding resource to unlocked, so other participants have the ability to lock this resource.

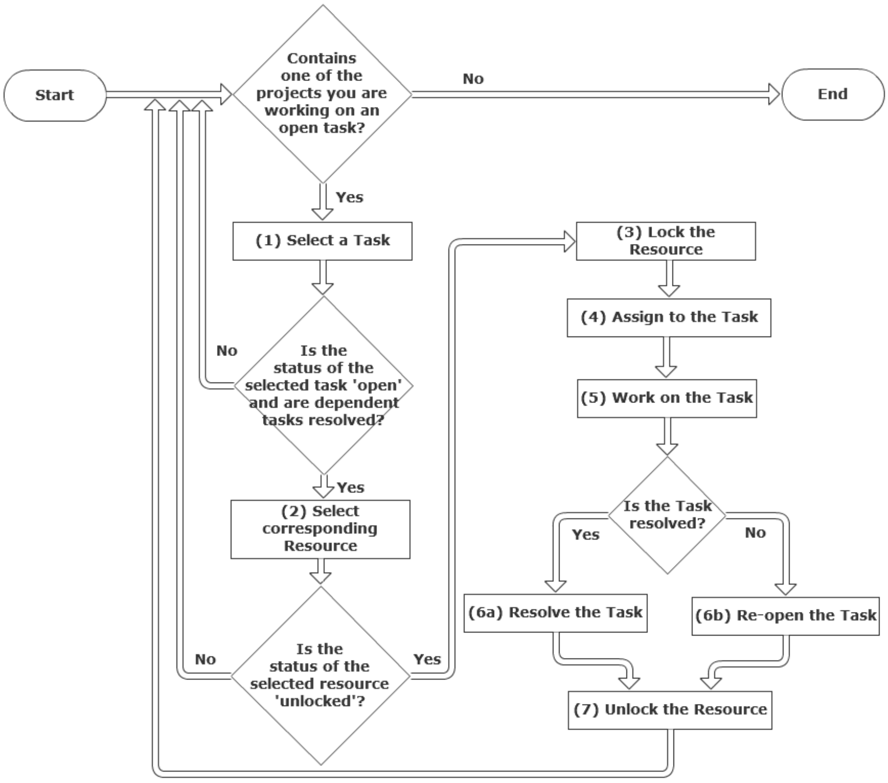

Start
Contains one of the projects you are working on an open task?
No
End
Yes
(1) Select a Task
No
Is the status of the selected task 'open' and are dependent tasks resolved?
Yes
(2) Select corresponding Resource
No
Is the status of the selected resource 'unlocked'?
Yes
(3) Lock the Resource
(4) Assign to the Task
(5) Work on the Task
Is the Task resolved?
Yes
No
(6a) Resolve the Task
(6b) Re-open the Task
(7) Unlock the Resource

**Figure 9.1:** *Process to successfully complete each project*

During this process participants encounter several collaboration challenges. These challenges arise from both the need to select a task of which all tasks it depends on are resolved, and the need to lock a task and corresponding resource. During the experiment participants of the control group have access to all information all the time, while participants of the test group only have access to contextualized information. As such participants of the test group have a more specific view of the available information.

The process is repeated until all 72 tasks of the experiment are completed. Then a questionnaire is distributed to all participants with additional questions to gather quantitative and qualitative data on their perceptions of speed, accuracy and usefulness of virtual office walls.

In the experiment we cannot use real software engineering work, because this would make the experiment too long to be feasible. Instead, we decided to use simple tasks. Additionally this also minimizes mistakes due to differences in software programming abilities of the participants. However, to accurately represent software engineering work we need tasks which are comparable to those encountered in a software engineering project. The tasks we ended up using are the modified fictional map tasks used by Espinosa et al. [Esp07], like adding a object to a map or adding an arrow between two objects. This type of task mimics important aspects of global software engineering teams including [Esp07]: (i) shared goals, (ii) interdependent activities and skills, (iii) the need for effective communication, and (iv) the need to articulate and interpret requirements correctly. These tasks have been used in previous experimental studies in the field of software engineering [Esp07, Sol10].

Furthermore, we decided to prevent communication between participants from the controlled experiment. Part of what would normally be communicated is replaced by the automated features of the technological solution used in the experiment, such as changing the current status of a resource. Other communication is prevented because it is likely to have a too strong impact on the outcome of the experiment. Allowing this communication would make it infeasible to independently measure the effect of the introduction of virtual office walls. When participants are unable to communicate directly with each other, all coordination actions take place in the environment in which the concept of virtual office walls is either implemented (test group) or not (control group). Therefore we modified the map tasks adopted by Espinosa et al. [Esp07] to remove the need for communication and to decrease the ambiguity of the requirements.

### 9.4.2 Dependent, Independent and Control Variables

In this section we discuss the three types of variables used in this experiment. An independent variable (factor) is a variable that is manipulated in the experiment. The values or settings for an independent variable are the test conditions. The impact of the different test conditions is measured by analyzing the dependent variables. Finally, there are circumstances that might influence a dependent variable but are not being investigated. These are called control variables and need to be controlled to limit the variability of the measures.

In this experiment only one variable is changed, the support environment and we defined two test conditions for this dependent variable: an environment in which (i) the concept of virtual office walls is enabled and (ii) the concept of virtual office walls is disabled.

Next to the independent variable we also defined 6 dependent variables:

- *Actual speed:* We measure the time (in seconds) it takes for participants to successfully complete each of the 72 tasks
- *Perceived speed* We measure the perceived speed by asking participants how they would grade the overall speed of work on a 5 point likert scale (very low, low, normal, high, very high) with a no-opinion option
- *Actual accuracy:* We measure the accuracy of the performed work by dividing the number of correct elements by the total number of elements
- *Perceived accuracy:* We measure the perceived accuracy by asking participants how they would grade the overall quality of the work on a 5 point likert scale (very low, low, normal, high, very high) with a no-opinion option
- *Ease of use of the system:* We measure the ease of use of the system by asking participants how they would grade the use of the system on a 5 point likert scale (very difficult, difficult, normal, easy, very easy) with a no-opinion option
- *Usefulness of the system:* We measure the usefulness of the system by asking participants how they would grade the overall support on a 5 point likert scale (very low, low, normal, high, very high) with a no-opinion option

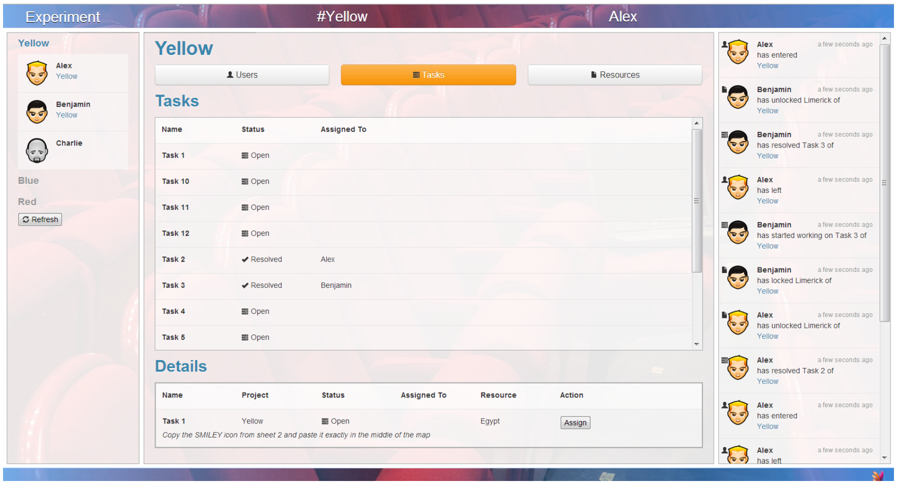
There also exist six control variables which need to be kept constant between the two groups of the experiment to avoid influencing the measurements: users, projects, maps, tasks, resources and tools. Finally, one randomized variable exist: the assignment of tasks to a participant, as such each user can complete an unequal number of tasks.

### 9.4.3 Tooling Environment

In this section, we discuss the environment we have used during the experiment to enable or disable the concept of virtual office walls.

To minimize the influence differences in design could have on the outcome of the experiment, we use the same environment for both groups. We only change whether or not the concept of virtual office walls is enabled. Figure 9.2 provides an overview of the environment used and in the remainder of this section each of the components will be shown and discussed in detail.

Firstly, we discuss differences in information shown about the team members of a user (the left section of the view). In figure 9.3a the information provided to the participants of the test group is shown and in figure 9.3b the information provided to the control group is shown. Comparing these two figures, it can been seen that participants in the control group have access to information about all



Yellow

Tasks

| Name | Status | Assigned To |
|---|---|---|
| Task 1 | Open | |
| Task 10 | Open | |
| Task 11 | Open | |
| Task 12 | Open | |
| Task 2 | ✓ Resolved | Alex |
| Task 3 | ✓ Resolved | Benjamin |
| Task 4 | Open | |
| Task 5 | Open | |

Details

| Name | Project | Status | Assigned To | Resource | Action |
|---|---|---|---|---|---|
| Task 1 | Yellow | Open | | Egypt | Assign |

*Copy the SMILEY icon from sheet 2 and paste it exactly in the middle of the map*

Alex a few seconds ago
has entered
Yellow

Benjamin a few seconds ago
has unlocked Limerick of
Yellow

Benjamin a few seconds ago
has resolved Task 3 of
Yellow

Alex a few seconds ago
has left
Yellow

Benjamin a few seconds ago
has started working on Task 3 of
Yellow

Benjamin a few seconds ago
has locked Limerick of
Yellow

Alex a few seconds ago
has unlocked Limerick of
Yellow

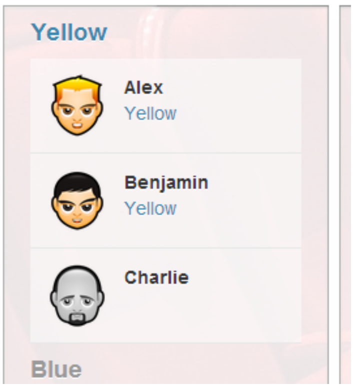
Alex a few seconds ago
has resolved Task 2 of
Yellow

Alex a few seconds ago
has entered
Yellow

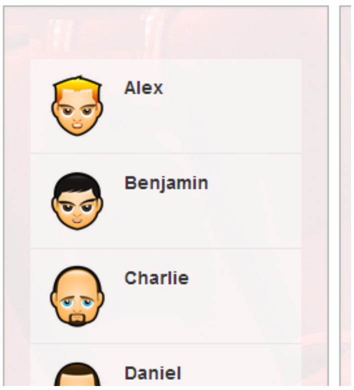
Alex a few seconds ago
has left

**Figure 9.2:** *Tooling environment*

other users, while participants in the test group only have access to information about the users of the project they are currently working on (project Yellow). This filter on active project makes it possible to enrich the visualization of the test group by showing which of the team members are currently working on the project. Because participants of the test group only have access to information related to their active project, a mechanism is needed to change this context. This can be done by selecting another project in the left section of the view (e.g. Blue). In this view only projects are shown in which you participate, all other projects are filtered out.

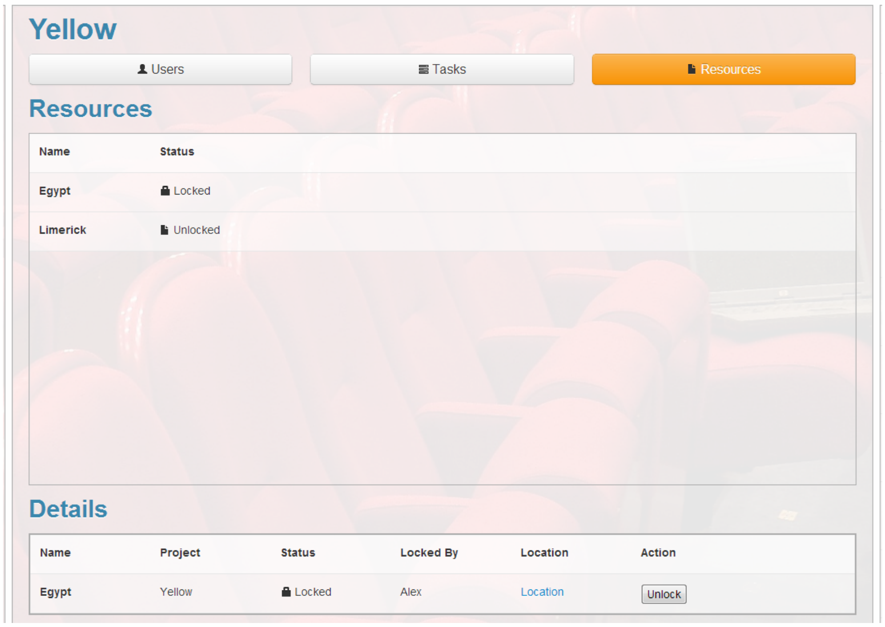


**(a)** *Test group*



**(b)** *Control group*

**Figure 9.3:** *Team members*



## Yellow

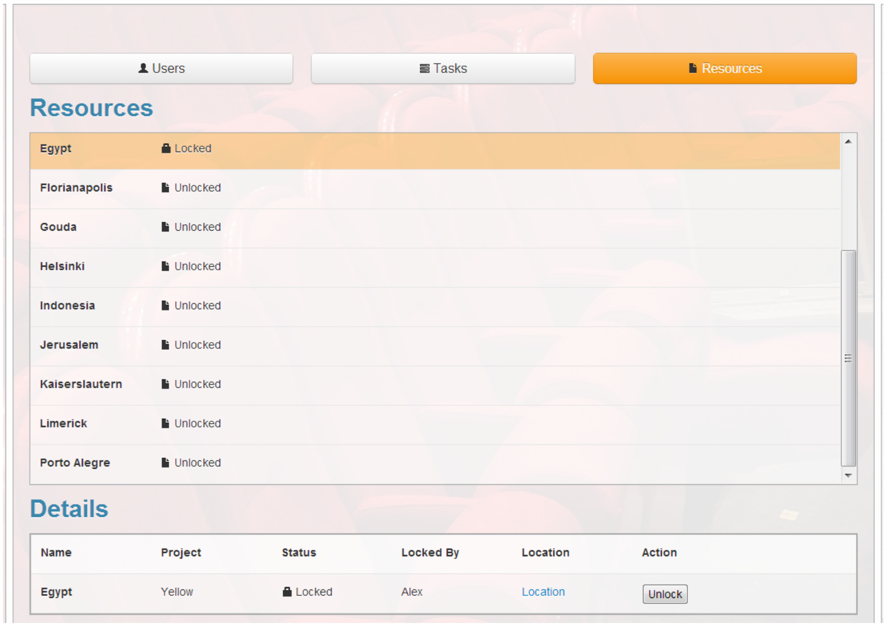
### Resources

| Name | Status |
|---|---|
| Egypt | Locked |
| Limerick | Unlocked |

### Details

| Name | Project | Status | Locked By | Location | Action |
|---|---|---|---|---|---|
| Egypt | Yellow | Locked | Alex | Location | Unlock |

(a) *Test group (with virtual office walls)*



### Resources

| | |
|---|---|
| Egypt | Locked |
| Florianapolis | Unlocked |
| Gouda | Unlocked |
| Helsinki | Unlocked |
| Indonesia | Unlocked |
| Jerusalem | Unlocked |
| Kaiserslautern | Unlocked |
| Limerick | Unlocked |
| Porto Alegre | Unlocked |

### Details

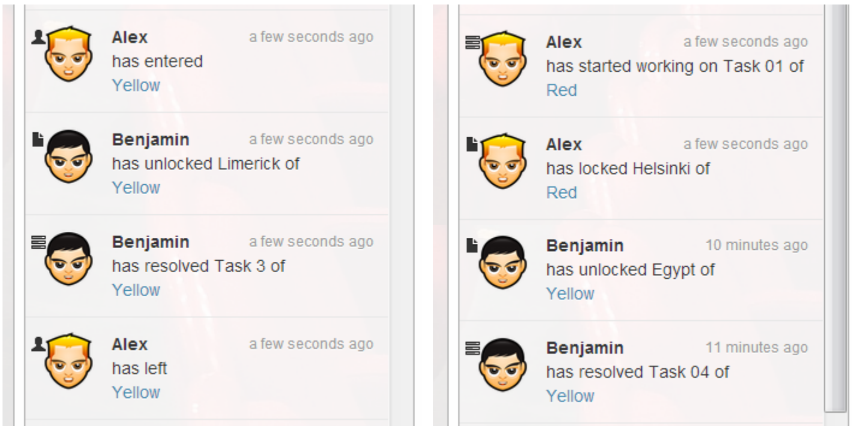
| Name | Project | Status | Locked By | Location | Action |
|---|---|---|---|---|---|
| Egypt | Yellow | Locked | Alex | Location | Unlock |

(b) *Control group (without virtual office walls)*

**Figure 9.4:** *An overview and detailed information*

Secondly, we discuss the middle section of the view in which an overview and detailed information about the users, tasks and resources is shown. Again, the main difference is that participants in the control group have access to information about all projects, while participants in the test group only have access to information about the project they are currently working on. This can be seen in figure 9.4. Figure 9.4a and 9.4b both show information about the status of resources. Comparing these two figures, it can been seen that participants of the test group only have access to information about the resources of the project they are currently working on, while participants of the control group have access to the resources of all projects.

Finally, we discuss the right section of the view in which information about the actions participants performed is shown (time-line). Again, participants in the control group have access to all actions, while participants in the test group only have access to actions related to the project they are currently working on. During the experiment participants perform actions to successfully complete the projects, for example locking a file and resolving a task. The following actions are shown on the time-line of both groups: locking a resource, unlocking a resource, start working on a task, stop working on a task and resolving a task. As a consequence participants in the control group get to see actions from a wide variety of projects (see figure 9.5b), while participant in the test group only see actions related to the project they are currently working on (see figure 9.5a). Next to the actions which appear on both time-lines participants of the test group also get to see actions of project members who either enter or leave the project space (by changing their active project), providing additional information about their active project.


Alex a few seconds ago
has entered
Yellow

Benjamin a few seconds ago
has unlocked Limerick of
Yellow

Benjamin a few seconds ago
has resolved Task 3 of
Yellow

Alex a few seconds ago
has left
Yellow

(a) *Test group*

Alex a few seconds ago
has started working on Task 01 of
Red

Alex a few seconds ago
has locked Helsinki of
Red

Benjamin 10 minutes ago
has unlocked Egypt of
Yellow

Benjamin 11 minutes ago
has resolved Task 04 of
Yellow

(b) *Control group*

**Figure 9.5:** *Actions performed by team members*

### 9.4.4 Context, execution and assignments

This study is conducted at IHomer, a Dutch software engineering company founded in August of 2008. The company currently is fully distributed, since the default work location of the employees is their home. As a consequence, all employees are experienced with dealing with the difficulties of developing software when working physically separated from each other. This distributed nature makes employees of this company particularly suitable as study participants for this experiment. Employees participated voluntarily in the experiment. In total 12 employees participated in the experiment, divided into two groups of 6 participants, one group which has access to an environment in which the concept of virtual office walls is enabled (test group) and one group which has access to an environment in which this concept is disabled (control group). The distribution of participants into these groups is random but balanced based on the seniority of the participants.

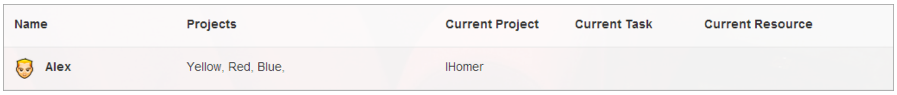
The experiment itself was conducted on a single day. During this day two runs were executed, one by the test group and one by the control group. The first run took place between 9:00 AM and 10:30 AM, in this run we used the environment in which the concept of virtual office walls was enabled (Test group). The second run took place in the afternoon, between 2:00 PM and 3:30 PM, in this run we used the environment in which this concept was disabled (Control group). To ensure participants of the control group were not influenced by participants of the test group, we asked them not to talk about the experiment until both runs were finished. As far as we know all participants did this. Because of the distributed nature of both the company and experiment we decided to execute both runs with participants working from their home. Before each run was executed, the participants were gathered in a Google Hangout[2] in which the objective of the experiment and the tasks were explained. Subsequently, a 15 minute demo was given to the participants by one of the researchers, in which the process of successfully completing a task was demonstrated. Finally, the participants were allowed to enter the environment and they were asked to leave the hangout (so they could not communicate with each other). When all projects were completed we again gathered all participants in a Google Hangout to thank them for their participation, and to distribute the questionnaire. During both runs, two of the authors were available to provide help if needed.

We conclude this section by giving an impression what actions are necessary to successfully complete each task. We show a scenario in which all actions of the process are demonstrated. In this discussion we provide a detailed description of all available information types, all necessary actions, and the tooling used to complete each action to complement the more general discussion in section 9.4.1.

[2]http://www.google.com/+/learnmore/hangouts

## Selecting a Task

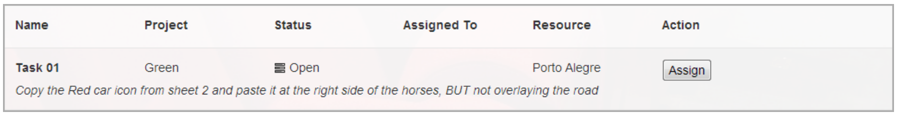
Before a participant is able to work on a specific task, he[3] first has to select an open task of one of the projects he is assigned to. To find out to which projects a participant is assigned, he should use the tooling environment discussed in the previous section. In this environment he can access all information about a user, see figure 9.6: (i) the name of the user, we changed the names of the participants to Alex, Benjamin, Charlie, Daniel, Ethan and Freddie to enable anonymous participation, (ii) the projects he is assigned to, a user is always assigned to three of the following projects: Blue, Green, Orange, Purple, Red and Yellow, (iii) the project he is currently working on, (iv) the task he is currently working on, and (v) the resource he is currently locking.



| Name | Projects | Current Project | Current Task | Current Resource |
|---|---|---|---|---|
| Alex | Yellow, Red, Blue, | IHomer | | |

**Figure 9.6:** *User details*

When a participant knows to which projects he is assigned, he can select an open task of one of these projects. Again the participant should use the tooling environment to determine the current state of the selected task. Each task is in one of the following three states: open, in progress or resolved. When the state of the task is resolved, the task is successfully completed and no further actions are required. When the state of the task is in progress, the task is currently being carried out by one of the participants. Finally, when the state of the task is open, work has to be carried out to complete this task. Next to the current state of the task also (i) the name, ranging from Task 1 through 12, (ii) the corresponding project, (iii) the description, (iv) the participant who is currently working on the task or the participant that has resolved the task, and (v) the corresponding resource are shown, see figure 9.7.



| Name | Project | Status | Assigned To | Resource | Action |
|---|---|---|---|---|---|
| Task 01 | Green | Open | | Porto Alegre | Assign |

*Copy the Red car icon from sheet 2 and paste it at the right side of the horses, BUT not overlaying the road*

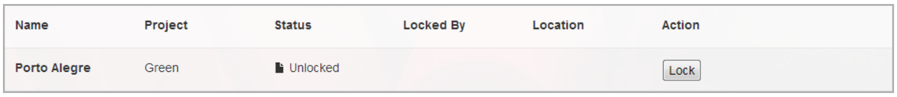
**Figure 9.7:** *Task details*

## Selecting a Resource

When a participant has selected an open task of which all tasks it depends on are resolved, he has to verify the status of the corresponding resource. A resource

[3]All participants of the experiment were male, so when talking about participants of the experiment we will use *'he'*

can only have two states: unlocked and locked. When the state of the resource is locked, another participant locked the resource and as a consequence the participant has to select another task. When the status of the resource is unlocked, the participant can lock the resource and start working on it. Next to the state of the resource also, (i) either the name of the city or country, (ii) the corresponding project, (iii) the participant who has currently locked the resource, and (iv) the location of the resource are shown, see figure 9.8.



| Name | Project | Status | Locked By | Location | Action |
|---|---|---|---|---|---|
| Porto Alegre | Green | Unlocked | | | Lock |

**Figure 9.8:** *Resource details*

### Locking a Resource

Now a participant has selected both an appropriate task and resource, he should use the environment to actually lock the resource indicating he is currently working on that resource.

### Assigning to a Task

When a participant has locked the resource belonging to the selected task, he should use the environment to actually assign the selected task to himself indicating he is currently working on that task.

### Working on a Task

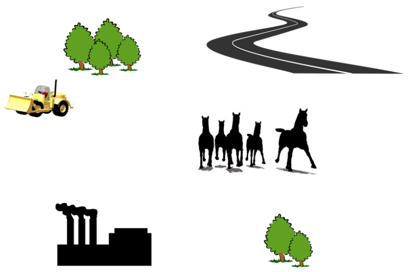
When a participant has both assigned himself to a task and locked the resource corresponding to this task, he is able to start working on the task. He first needs to download the resource from a central repository to acquire the file on which he needs to work. In this experiment we use Dropbox[4] as repository, because of its easy to use web interface and because it enables a history of all versions of a single file. Secondly, the participant has to edit the file (resource) he just downloaded either by using PowerPoint[5] or Impress[6]. Each file consist of two parts, a map and a set of objects (see figure 9.9). By adding objects and arrows to the map, according to the description of the current task, a modified version of the file is created. In contrast to the map requirements of Espinosa et al. [Esp07] we more explicitly defined the tasks and tried to remove ambiguity from them since inter-participant communication was not allowed to resolve such issues. Therefore we decided on the following categories of task instructions:

[4]`http://dropbox.com`
[5]`http://office.microsoft.com/powerpoint/`
[6]`http://libreoffice.org/impress/`

1. *Adding an object:* Tasks in this category include descriptions to copy and replace an object from the set of objects to the map. For example: *"Copy the telephone object from the set of objects to the map and paste it directly above the factory"*

2. *Drawing an arrow:* Tasks in this category include descriptions to add an arrow to the map. For each arrow multiple properties are specified like, the starting point, ending point, color, weight, and style of the arrow, either solid or dashed. An example of such a task description is: *"Draw a green dashed arrow of width 5pt from the telephone object to the most right horse"*.

3. *Copying and moving an arrow:* Tasks in this category include descriptions to copy and move an arrow already drawn on the map e.g. *"Copy and paste the green arrow and move it, starting at the most right horse to the left of the bulldozer"*.


**(a)** *Map*

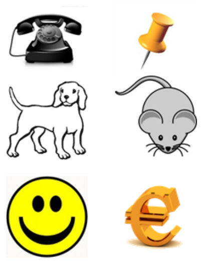

**(b)** *Control group*

**Figure 9.9:** *Set of Objects*

Each task only includes a single task instruction. Finally, when the participant has completed the current task, see figure 9.10 for an example of a modified resource, he needs to save and upload the file to the repository.

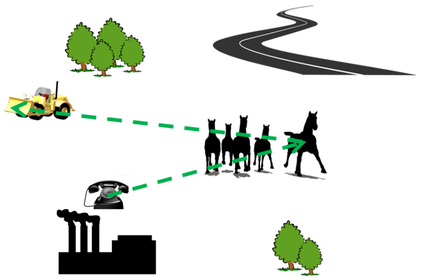


**Figure 9.10:** *Example of a modified resource*

#### Resolving or re-open a Task

When a participant has completed the task to which he assigned himself, he should update the status of the task to resolved. However, if the participant has not completed the task and decides to work on another task, he should re-open the task so someone else is able to resolve it.

#### Unlocking a Resource

When a participant has either resolved or re-opened the task to which he assigned himself, he should also update the status of the resource to unlocked, so other team members can lock this resource.

## 9.5 Findings

In this section the main findings of this study are presented. We discuss the six research questions regarding the accuracy, speed and usefulness of virtual office walls. For each of these domains we will present quantitative results from the experiment and questionnaire, and qualitative results from the questionnaire. The results of the data are visualized in different ways: we use (i) histograms to illustrate the distribution of the data, and (ii) stacked bars without the neutral data to illustrate the ratio between the positive and negative responses. Finally we also evaluate the hypotheses based on the findings of the experiment and questionnaire.

### 9.5.1 Speed of the Work Performed

We measured the influence of virtual office walls on the actual speed of work by measuring the time it took the participants to successfully complete all projects of the experiment. The control group needed 39 minutes to complete these projects, while the test group only needed 35 minutes to complete their work. By comparing these results, it can be seen that the introduction of virtual office walls increases the speed of work by almost 10%. Since we also measured the time participants needed to perform their tasks (so actually performing the instruction of the task), we can also calculate the time they needed to coordinate their work. The control group needed 19 minutes to coordinate their work while the test group only needed 15 minutes. This difference makes the impact of the introduction of virtual office walls even more obvious, since the time needed to coordinate work was decreased by approximately 20%.

The results of the questionnaire were used to analyze the perception of the speed of work. We asked each participant to grade the speed of the work performed by himself, the work performed by his team members, and the work performed by the whole team, see figure 9.11. The results of the test group were slightly more positive than the results of the control group. Since participants in the test

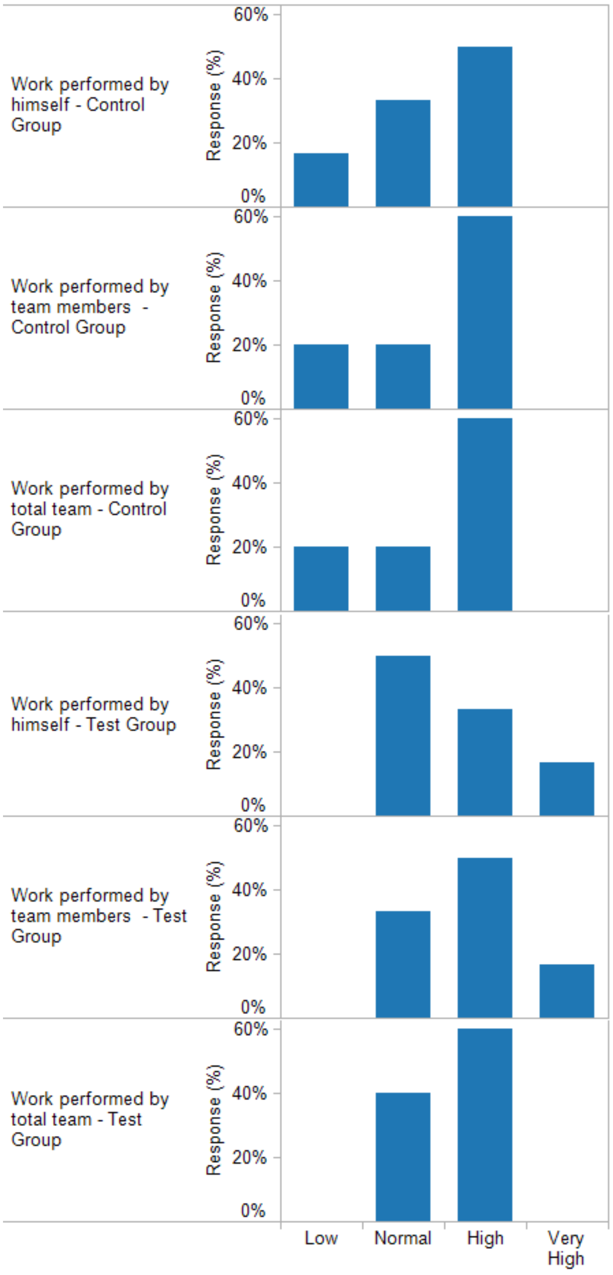


**Figure 9.11:** *Perception of the speed of work*

group have not reported any negative values at all and some of them indicated the perception of speed to be very high. Participants of the control group, however, reported several negative values and none of them rated the perception of speed very high. One of the participants of the control group gave as feedback: *[Charlie] "It seemed that I was searching a lot for a task that had an unlocked file associated to it. My team members were locking files that I needed to edit"*.

However, we cannot accept or reject our hypotheses regarding the speed of work because of the small sample size of the experiment and the individual differences in performing the specified tasks. We have, however, some indications that the introduction of virtual office walls has decreased the time needed to coordinate work and as such has a positive impact on the total speed of work (**H1**). Additionally, we found some indicators corresponding to the hypothesis that virtual office walls have a strong positive impact on the perception of the speed of work (**H2**).

## 9.5.2 Accuracy of the work performed

Next to measuring the speed of work, we also measured the accuracy. The metric used to verify the accuracy of the work performed, is dividing the number of correct elements by the total number of elements. Both the test group and the control group have performed their tasks without making a mistake. As such, both groups have an accuracy of 100%. The removal of ambiguous requirements from the original experiment appears to be the reason for this. Note that this was done to control communication by excluding it from the experiment.

The perception of the work accuracy, however, cannot be derived by analyzing the results of the performed tasks. Therefore, we asked each participant to grade the accuracy of the work performed by himself, the work performed by his team members, and the work performed by the whole team. From these results it can be seen that in general the control group is more negative about the accuracy of the work performed than the test group, see figure 9.12.

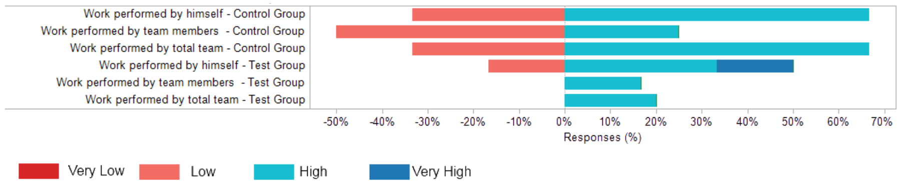


**Figure 9.12:** *Perception of the work accuracy*

These findings correspond with both hypotheses regarding accuracy: (**H3**) the introduction of virtual office walls has no impact on the accuracy of the work carried out, and (**H4**) the introduction of virtual office walls has a strong positive impact on the perception of accuracy of the work carried out. Again, the sample size of the experiment is too small to either reject or accept these hypotheses.

What is perhaps most striking about the data we gathered about the perception of work accuracy, is the difference in the number of *No-Opinions* between these groups. Five participants of the control group have indicated they were not able to grade the accuracy on a scale ranging from very low to very high, against only one participant of the test group. Several of the participants of the control group explain their choice, for example: *[Ethan] "I was unable to grade the work that my team had done, since I did not know what their assignments were, or what the goal was"* and *[Freddie] "I have not checked the accuracy of my team members, I opened the file to execute the task but I have not checked back at older tasks to see if the file reflects the right state"*. This is striking because it indicates participants of the control group have a low level of awareness about the accuracy of work carried out.

### 9.5.3 Usefulness of Virtual Office Walls

Finally, we measured both the usefulness and the ease of use of virtual office walls. To analyze the usefulness of virtual office walls we asked the participants of the test group to grade the usefulness of (i) the integration of the information, (ii) being able to see the current context of your project members and the project they are currently working on, and (iii) the differentiation between information that was helpful to your current task and information that was not. We only considered the test group in this analysis, because this group has actually experienced working with virtual office walls and as such is able to give a well-argued opinion of the usefulness of this concept. The results of these opinions (see figure 9.13), provide strong indications that context based filtering of information is useful, since most participant graded the usefulness in the range normal to very high.

Next to the usefulness of virtual office walls we also asked participants of both groups to grade how easy or difficult it was to (i) see the active project of their team members and (ii) differentiate between information that was helpful to their current task and information that was not. The outcomes of these questions are depicted in figure 9.14. It is obvious that participants in the test group find it much easier to acquire the active context of their team members and to differentiate between relevant and non-relevant information. One of the participants of the control group, Daniel, indicated he had the feeling there was a *"Lock race"* going on as there was no way to determine who would carry out which task on what resource.

These findings correspond with the hypotheses regarding the usefulness and the ease of use of virtual office walls. Because, experienced software engineers indicated they considered the introduction of virtual office walls to be useful (**H5**). Participant also indicated that it was easy to differentiate between information that is relevant and information that is not, and that the introduction of these walls makes it easier to understand which other project members are working at the same time on the same project (**H6**).

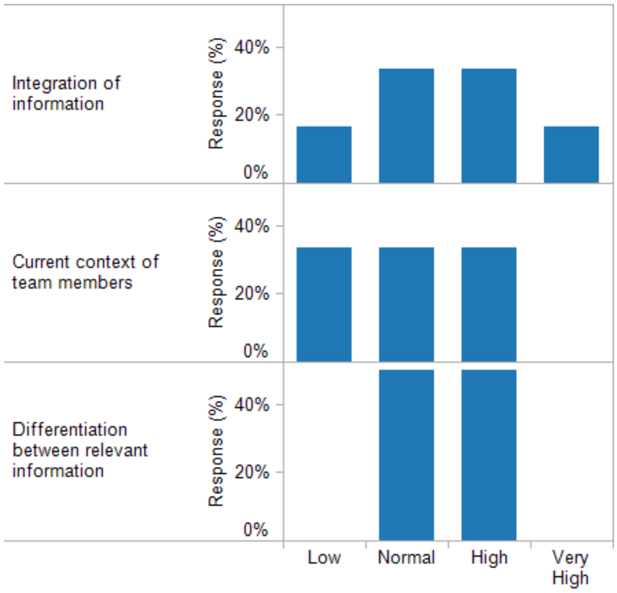


**Figure 9.13:** *Usefulness of virtual office walls*

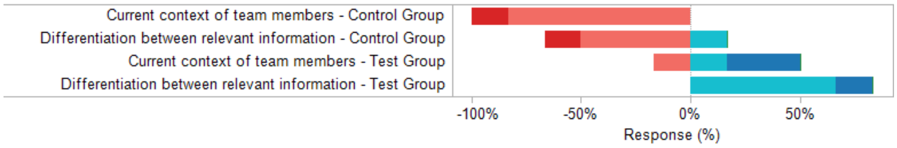


**Figure 9.14:** *Ease of use of virtual office walls*

## 9.6 Threats to Validity

In this section we discuss the threats to validity for this experiment on four aspects: construct validity, internal validity, external validity and conclusion validity.

Construct validity regards the extent in which the variables measured actually measure the constructs of interest. The main threat to construct validity of this experiment is that we use the modification and manipulation of PowerPoint maps instead of real programming work. One could argue that our study does not investigate distributed software engineering, but distributed PowerPoint editing. However, by doing this (i) internal validity is increased by eliminating differences in individual programming skills, and (ii) throughput time is reduced. Furthermore, while on the one hand we were able to measure the speed by checking the time stamps of the events in the repository and the accuracy by comparing the outcomes to the predefined results directly. On the other hand we have to ask the participants for their opinion to measure the perception of speed, accuracy and usefulness. One could argue that in general it is difficult to measure opinions. As such we used simple metrics on a 5-point likert scale and asked directly for the perceived effects.

Internal validity is especially relevant in studies, such as this, that attempt to identify a causal relationship. The question regarding internal validity is whether the observed effect was actually caused by the researched factor. In this experiment we managed to keep the experimental conditions relatively stable, considering we undertook the experiment in a real-life company. However, we need to express that the tool was demonstrated to the engineers before. As such this previous experience (without the virtual office wall functionality) might have influenced their opinions. Furthermore, the control group and the test group consisted of different people which can have influenced the results as well.

External validity is of interest in studies that want to draw generalized conclusions. Although, experiments are in general highly externally valid [Woh00], for this specific study it is too early to make generic claims. First of all, the sample size is too small to draw statistically significant conclusions. Second of all, because the type of work the participants performed (placing and moving objects on a map) and the average duration of the tasks only an approximation of the actual work of software engineers is given.

Finally, conclusion validity expresses the extent in which the intervention (providing the virtual office walls) actually led to the observed outcome, and as such questions the reliability of the studies' conclusions. We acknowledge that the small number of samples and data points does not allow us to draw causal conclusions. This is however, a known problem, in experimental studies in software engineering teams [Cal12].

## 9.7 Concluding Remarks

## 9.8 Conclusions

We conducted this research to discover how valuable virtual office walls are to distributed software engineers. To reach this goal we performed a controlled experiment. In this experiment we looked at the accuracy and speed of the work performed and perception of these. Furthermore we investigated the extent in which experienced distributed software engineers consider virtual office walls to be useful.

The main findings of the study are the following:

1. Virtual office walls make work coordination easier, because they assist in differentiating between information that is relevant to the current activity of an engineer and information that is not

2. Virtual office walls contribute to an increased perception on overall performance.

Even though, the data we gathered is not statistically significant due to the small sample size, it is interesting to see how these two main benefits are related to each other. This is more clearly shown in figure 9.15.

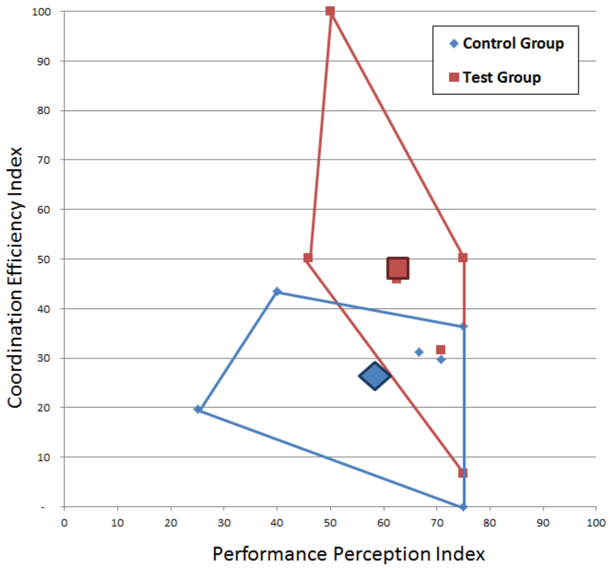


**Figure 9.15:** *Coordination - Performance system*

In this figure each data point represents the results in the experiment of one of the participants along two axes: the *'coordination efficiency index'* on the y-axis, the *'performance perception index'* on the x-axis. The *'coordination efficiency index'* is the normalized time the participant took to perform the coordination portion of the experiment. This value is normalized to give the most efficient participant the score '100' and the least efficient participant the score '0'. The *'performance perception index'* is the normalized performance perception the participant reported on the 5-point Likert scale. This value is normalized so the maximum possible score is '100' (6 times very high) and the minimum score is '0' (6 times very low). Furthermore, for each of the two groups the outliers are connected to create a tetragon to illustrate the space it encloses. Finally, the average values, roughly at the center of the tetragons, are depicted as a fat data point. In the figure it can be seen that the test group overall scores higher than the control group on both metrics. It will be interesting to investigate this with a much larger sample size to see if this conclusion will continue to hold.

Furthermore, the data collected in the questionnaire following the experiment also indicated the overall usefulness of the virtual office wall concept. In general, distributed software engineers are overloaded with an abundance of information, such as information available on their systems, mailbox and tools, so every improvement in helping them cope will be welcomed with open arms. And doesn't that hold true for all of us? Imagine a world in which our e-mail would configure itself to only include mails related to our current task. What if our computers would notify us if our colleague is editing the same code as us or recommend us who to ask for help when we are struggling with something? Wouldn't our lives be much less complicated if our working environment would adapt itself to us, instead of the other way around?

This is the promise of mechanisms such as virtual office walls and with the study presented in this chapter we have shown indications there is value in pursuing this further. The next step of this research should be a prioritization of the information available to a software engineer, to be able to construct virtual office walls which regulate information based on both the current activity of a software engineer and the importance of the information.

### 9.8.1 Virtual Office Implications

In this chapter we have reported on a controlled experiment to study whether there is a relation between the presence of virtual office walls and the actual and perceived speed and accuracy of the work carried out by the participants. Further, we also measured the extent in which the participants experience the presence of virtual office walls as useful. As such this chapter contributes to answering the fifth research question of this dissertation:

**Research Question 5**

*What is the value of automating the process of restricting the available information to that information a software engineer needs to carry out his current activity*

This controlled experiment provided us with useful insights of the value of virtual office walls for global software engineers during their day to day activities. However, we could not derive new requirements of a virtual office.

# Part V

# Epilogue

Chapter 10

# Requirements of a Virtual Office

In this chapter we summarize the requirements of a virtual office we identified. The main goal of this dissertation is *"To support global software engineers with technological support for aiding them to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities"*. While we do not yet have an all-encompassing approach to reach this goal, we did study three aspects of the design and creation of a *'Virtual Office'*. First, we studied what approach should be used to research how best to support awareness in global software engineering. Secondly, we looked at the value of communication in global software engineering. Finally, we researched the information needs of global software engineers.

## 10.1 Constructing a Virtual Office

Firstly, we discuss the requirements related to constructing a virtual office. These requirements are based on our vision on how to provide distributed software engineers with a sufficient level of awareness for their work activities.

We first compared the co-located setting with the distributed setting and concluded that in a co-located setting awareness is spread relatively passively and unobtrusively while it takes more effort in a distributed setting. In a distributed setting a mechanism is needed which automatically regulates information based on the current activity of an engineer, a *'Virtual Office Wall'*. Such a mechanism can help distributed software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness. Next, we discussed the two prerequisites which should be fulfilled to construct virtual office walls. The first prerequisite concerns having access to a data set which at least contains the required data at a certain time. The second prerequisite of a virtual office wall concerns a method to differentiate between required and not required information. The requirements of a virtual office we have derived based on this analysis are:

**Req 1. Facilitate acquiring awareness relatively passively and unobtrusively**
*Software engineers should be able to acquire a sufficient level of awareness for their work activities in a relatively passive and unobtrusive fashion, similar to a co-located setting. Therefore the analytical process of accessing, combining and filtering information should be supported to avoid misunderstandings, inconsistencies, incompatibilities and duplicated information.*

**Req 2. Facilitate having access to a data set which at least contains the required data to carry out work activities**
*Software engineers need a wide variety of information to carry out their work activities. For example information about the requirements and the software repository.*

**Req 3. Facilitate combining data from different sources**
*Software engineers need to combine and integrate information from different sources to create valuable information. For example combining information about the artifacts that are usually modified together and information who most frequently modified the related source code files, makes it easier for a software engineer to determine who best to contact in case of doubt.*

**Req 4. Facilitate differentiating between required and not required information**
*Software engineers should be able to differentiate between required and not required information to regulate the available information to that information they need to carry out their current work activity.*

**Req 5. Facilitate a valid representation of the context of a software engineer**
*Software engineers need a way to represent their context to determine which information is required while performing a work activity.*

After identifying the first five requirements on how to provide distributed software engineers with a sufficient level of awareness we presented our approach to develop and validate such solutions. This iterative approach consists of three parts (i) identify real-life global software engineering problems, (ii) propose and implement solutions for these problems, and (iii) evaluate these solutions in an industrial setting.

## 10.2 Communicating in a Virtual Office

Secondly, we discuss the requirements of communicating in a virtual office. These requirements are either related to the overhearing of conversations of others, or to the sharing of information and corresponding emotions. First we provided a theoretical motivation why the overhearing of conversations of others is valuable to a distributed software engineering team. We provided a definition of a conversation, discussed the various uses conversations have in collaborative work, and provided a definition of an *'Open Conversation Space'*. Next, we presented an empirical study in which we evaluated the value of overhearing conversations in the field of software engineering. In this empirical study we investigated whether research on how to enable overhearing conversations in a distributed setting is worth pursuing. We identified: (i) the benefits and challenges of having insight in active conversations, (ii) the important types of information about a conversation, (iii) the actions possible on a conversation, and (iv) the benefits and challenges of having access to the finished conversations. For each of these benefits, challenges, information items and possible actions we determined their relative importance. Finally, we presented a technological implementation which enables the overhearing of conversations in a distributed setting and performed an empirical case study to measure the value of overhearing conversations in global software engineering.

Based on the contributions and the lessons learned of this line of research we have derived the following requirements of a virtual office:

**Req 6. Facilitate starting conversations**
*Software engineers should be able to have conversations, therefore it should be possible to initiate a conversation. For example by choosing a specific person, or a specific group of people to initially participate in the conversation.*

**Req 7. Facilitate detecting active conversations**
*Software engineers should be able to overhear conversations of others, therefore it should be possible to find out about active conversations. Software engineers could, for example, detect a conversation because they hear or see people talk to each other.*

**Req 8. Facilitate monitoring active conversations**
*Software engineers should be able to access information about the conversation without actually joining it.*

**Req 9. Facilitate participating in conversations**
*Software engineers should be able to become a participant in a conversation. They can, for example, become a participant in a conversation because they are invited into an ongoing conversation or because they actively joined a conversation they overheard.*

**Req 10. Facilitate finishing conversations**
*Software engineers should be able to finish a conversation.*

**Req 11. Facilitate having conversations which cannot be overheard by others**
*Software engineers should be able to have conversations of a private nature, which cannot be overheard by others.*

**Req 12. Facilitate changing the degree of involvement in a conversation**
*Software engineers should be able to change how aware they are of a conversation by changing their degree of involvement. They should, for example, be able to change their involvement from actively listening to participating in a conversation.*

**Req 13. Facilitate having insight in the finished conversations**
*Software engineers should be able to find out about a finished conversation to access information about it.*

After measuring the value of overhearing conversations, we also measured the value of microblogging with mood-indicators (MBMI) in global software engineering. We collected the empirical data needed, by mining over a year of usage data of such a microblogging system. In this study we researched (i) the topics discussed in a MBMI, (ii) the impact of the introduction of a MBMI on a software team, (iii) the impact of the distribution on the use of a MBMI, and (iv) how team composition impact collaboration with a MBMI. Based on the results of this content analysis we interviewed five distinctive users of the system. In these interviews we asked questions about what was unclear to us in the analysis and we asked follow-up questions we had based on this analysis.

Based on this empirical study we have derived the following requirements of a virtual office:

**Req 14. Facilitate sharing information about a new topic**
*Software engineers should be able to share information about a new topic to stay current on each other's experiences and latest exploits.*

**Req 15. Facilitate responding to an existing topic**
*Software engineers should be able to respond to an existing topic to add information or to answer a question.*

**Req 16. Facilitate sharing mood**
*Software engineers should be able to express their mood, for example by specifying a happiness score ranging from totally unhappy to totally happy.*

## 10.3 Information Needs in a Virtual Office

Finally, we discuss the requirements of information needs in a *Virtual Office*. We conducted an Estimate-Talk-Estimate study with experienced software engineers on how to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer. Based on this study we identified the following requirements:

**Req 17. Facilitate informing software engineers immediately of information about completed artifacts**
*Software engineers are especially interested in immediate updates of information about completed artifacts, e.g. requirements, design, and verification results.*

**Req 18. Facilitate informing software engineers immediately of information about the technological solution**
*Software engineers are especially interested in immediate updates of information about the technological solution itself, e.g. the selected solutions, the design of the components, and the design of the interfaces.*

**Req 19. Facilitate interrupting software engineers to provide them with new information**
*Software engineers should be interrupted to immediately provide them with information they want to know immediately.*

**Req 20. Facilitate controlling the moments during which one prefers not to be interrupted**
*Software engineers should be able to control whether they prefer to be or prefer not to be interrupted, especially during activities of a high interactive nature and activities which require a high level of concentration.*

Lastly, we conducted a controlled experiment with experienced software engineers as study participants. In this experiment we researched the value of virtual office walls in global software engineering. This experiment provided us with useful insights of the value of virtual office walls for global software engineers, however we could not derive new requirements of a virtual office.

# Chapter 11

# Conclusion

In the studies presented in this dissertation we researched aspects of the design, implementation and evaluation of a virtual office. These studies all contribute to the main goal of this dissertation:

**Research Goal**

> *"To support global software engineers with technological support for aiding them to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities"*

While we do not yet have an all-encompassing approach to reach this goal, we did study three important aspects of the design and creation of a *'Virtual Office'*.

First, we studied what approach should be used to research how best to support awareness in global software engineering. We have focused on the differences between the co-located setting and the distributed setting and concluded that a mechanism is needed which automatically regulates information based on the current activity of an engineer, a *'Virtual Office Wall'*. Furthermore, we provided an approach to develop and validate such a solution. Secondly, we looked at the value of communication in global software engineering. We researched both the value of overhearing conversations, and the value of microblogging with mood indicators in such a setting. These studies provide empirical evidence that both overhearing conversations of others, and microblogging with mood indicators are valuable to distributed software engineering teams. Thirdly, we researched the information needs of global software engineers. We presented two lists: a list of information items of which software engineers want to be informed immediately, and a list of activities during which software engineers prefer not to be interrupted. In addition to these lists, we also provided a look-up table which can be used to determine whether or not software engineers want to be immediately informed of an information item, even though they are performing an activity during which they prefer not to be interrupted. We concluded this part by studying the value of automating the process of restricting the available information to that information a software engineer needs to carry out his current activity and concluded that such mechanisms contribute to an increased perception on overall performance and that they make work coordination easier. Finally, we derived a set of requirements a virtual office should fulfill. These requirements contribute to the main goal of this dissertation, because they provide important guidelines on how best to provide global software engineers with the information they need.

In the remainder of this chapter we give a summary of the core contributions of this dissertation, revisit the research questions of this dissertation, and evaluate the research presented in this dissertation.

## 11.1 Summary of Contributions

All chapters in this dissertation have individual contributions. We summarize the core contributions below:

### General Contributions

- The analysis and identification of a set of requirements of a virtual office *Chapter 2, Chapter 4, Chapter 7 and Chapter 8*

- Empirical evidence that the value of awareness sharing technology is higher for people that work more distributed from their team and is higher when a larger portion of the team uses it *Chapter 6 and Chapter 7*

- Empirical evidence that in settings where part of the team works co-located the use of awareness sharing technology should be stimulated to encourage its use by people that work mostly co-located *Chapter 6 and Chapter 7*

### Constructing a Virtual Office

- The identification of the concept of virtual office walls *Chapter 2*

- Empirical evidence that automatically erecting virtual office walls have the potential to provide distributed software engineers with the context of their current activity *Chapter 2*

- The description of an iterative approach to research how to provide distributed software engineers with a sufficient level of awareness, in which global software engineers and researchers collaborate to (i) identify real-life problems, (ii) propose and implement solutions for these problems, and (iii) evaluate the solutions for these problems *Chapter 3*

### Communicating in a Virtual Office

- The identification of the concept of a conversation in the context of global software engineering *Chapter 4*
- The identification of the concept of an open conversation space *Chapter 4*
- The analysis and identification of the challenges and benefits of an open conversation space *Chapter 5*
- Empirical evidence that an open conversation space is a valuable concept for software engineering teams *Chapter 5*
- The design and implementation of a virtual open conversation space: Communico *Chapter 6*
- Empirical evidence that a virtual open conversation space is a valuable concept for global software engineering teams *Chapter 6*
- Empirical evidence that the introduction of a microblogging solution with mood indicators in a global software engineering environment increases team-connectedness and eases access to information that is traditionally harder to consistently acquire *Chapter 7*

### Information Needs in a Virtual Office

- Empirical evidence that software engineers want to be immediately informed of a wide variety of information, especially of information regarding (i) completed artifacts, and (ii) the technological solution itself *Chapter 8*
- Empirical evidence that software engineers prefer not to be interrupted when they are performing activities (i) of a highly interactive nature, and (ii) which require a high level of concentration *Chapter 8*
- The indication that software engineers do not want to be immediately informed of important information when they are performing an activity during which they prefer not to be interrupted *Chapter 8*
- Empirical evidence that virtual office walls make work coordination easier *Chapter 9*
- Empirical evidence that virtual office walls contribute to an increased perception on overall performance *Chapter 9*

## 11.2 Revisiting the Research Questions

We have conducted several studies to determine (i) what approach should be used to research how best to support awareness in global software engineering, (ii) the value of communication in global software engineering, and (iii) the information needs of global software engineers. In this section we attempt to answer the research questions of each of these three aspects.

### Constructing a Virtual Office

**Research Question 1**

*What are the requirements for technological support to provide distributed software engineers with the context of their current work activity?*

To answer the first research question, we have formulated both our vision on how to provide distributed software engineers with a sufficient level of awareness and an approach to implement such a solution. We have formulated our vision in chapter 2. We first compared the co-located setting with the distributed setting and concluded that in a co-located setting awareness is spread relatively passively and unobtrusively, while it takes more effort in a distributed setting. Therefore, in a distributed setting, we proposed the use of a mechanism which has the potential to regulate the available information to that information an engineer needs to carry out his current activity: a virtual office wall. Next, we discussed the prerequisites of such a construct: (i) access to a data set which at least contains the required data at a certain time, and (ii) a method to differentiate between required and not required information. Finally, we discussed a specific method to fulfill these prerequisites and validated this application in a practical case setting as well.

After formulating our vision on how to provide distributed software engineers with a sufficient level of awareness, we presented our approach to develop, implement and validate such solutions in chapter 3. In this chapter we described an iterative approach which consists of three parts: (i) identify real-life global software engineering problems, (ii) propose and implement solutions for these problems, and (iii) evaluate these solutions in an industrial setting. Next, we discussed that it is highly desirable to collaborate with companies in which distributed collaboration is common. This is because the people working at such a company experience the difficulties of working distributed from each other on a daily bases and therefore have a good understanding of what is needed to improve this situation. Another advantage of this collaboration is the ability to perform high quality evaluations because such a company perfectly matches the target setting for which we are attempting to solve issues. Finally, we discussed the process we use to design and implement the solution itself and discussed the three requirement such a process should fulfill: (i) being able to cope with uncertainty and changing requirements,(ii) involving the intended users of the system as soon as possible, and (iii) stimulating the usage of the proposed solution by all users.

Both our vision on how best to support global software engineers, and our approach to develop, implement and validate technological solutions which fulfill this vision are important aspects in answering the first research question. We used this vision in answering the other research questions and in discussing the findings. Therefore, after conducting these empirical studies, we were able to derive requirements a virtual office should fulfill. These requirements are presented after each chapter of this dissertation, and each requirement is related to either (i) constructing a virtual office, (ii) communicating in a virtual office, or (iii) information needs in a virtual office. An overview of all the requirements is given in chapter 10. These requirements contribute to the first research question, because they provide important guidelines on how best to design and implement a virtual office.

### Communicating in a Virtual Office

**Research Question 2**

*What is the value of overhearing conversations in global software engineering?*

The second research question is answered by (i) providing a theoretical motivation why the overhearing of conversations of others is valuable to a distributed software engineering team (Chapter 4), (ii) evaluating the value of the concept of overhearing conversations in the field of software engineering (Chapter 5), and (iii) evaluating the value of overhearing conversations in global software engineering from actual industrial experience (Chapter 6). In chapter 4 we provided a definition of a conversation, discussed the reasons for having conversations, provided a definition and requirements of an open conversation space, and discussed the advantages of overhearing the conversations of others. Next, in chapter 5, we presented an empirical study in which we evaluated the value of overhearing conversations in the field of software engineering. The results of this study include (i) the benefits and challenges of having insight in active conversations, (ii) the important types of information about a conversation, (iii) the actions possible on a conversation, (iv) the benefits and challenges of having access to the finished conversations, and (v) the relative importance of these benefits, challenges, information items and possible actions. From this study we concluded that an open conversation space is valuable to software engineering teams and that research about support for overhearing conversations in global software engineering is worth pursuing. Finally, in chapter 6, we presented Communico, a technological implementation which enables the overhearing of conversations in a distributed setting and explained how this solution fulfills the requirements of an open conversation space. We used this solution to perform an empirical case study to measure the value of overhearing conversations in global software engineering from actual industrial experience. Based on the findings of this empirical study we concluded that being able to overhear conversations of colleagues in a global software engineering team is valuable.

**Research Question 3**

*What is the value of microblogging with mood-indicators in global software engineering?*

To answer the third research question, we mined over a year of usage data of a microblogging with mood-indicators system (MBMI). In this study, presented in chapter 7, we researched (i) the topics discussed in a MBMI, (ii) the impact of the introduction of a MBMI on a software team, (iii) the impact of the distribution on the use of a MBMI, and (iv) how team composition impact collaboration with a MBMI. Based on the results of this content analysis we interviewed five distinctive users of the system to clarify what was unclear to us in the analysis and to ask them follow-up question we had based on this analysis. This empirical study provided empirical evidence that the introduction of a microblogging with mood-indicators solution in a distributed software engineering team increases team-connectedness and eases access to information that is traditionally harder to consistently acquire.

### Information Needs in a Virtual Office

**Research Question 4**

*How to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer?*

To answer research question four we conducted an Estimate-Talk-Estimate study with experienced software engineers on how to regulate information available to software engineers based on both the importance of that information and the current interruptibility of the engineer. We structured this research, presented in chapter 8, using the five engineering process areas defined in the CMMI for development: Requirements Development, Technical Solution, Verification, Validation, and Product Integration. For each of these areas we determined (i) what information software engineers want to know immediately, and (ii) during which activities software engineers prefer not to be interrupted. These classifications resulted into two lists: a list of information items of which software engineers want to be informed immediately and a list of activities during which software engineers prefer not to be interrupted. The outcomes of this study introduced a contradiction, since participants on the one hand indicated that they like to be informed immediately of several information items. On the other hand they also indicated that they prefer not to be interrupted during some activities. Therefore, we also provided a look-up table which can be used to determine whether or not software engineers want to be immediately informed of an information item, even though they are performing an activity during which they prefer not to be interrupted.

**Research Question 5**

*What is the value of automating the process of restricting the available information to that information a software engineer needs to carry out his current activity?*

The answer of the final research question of this dissertation is discussed in chapter 9. We performed a controlled experiment with experienced software engineers as study participants to study whether there is a relation between the presence of virtual office walls and the actual and perceived speed and accuracy of the work carried out by the participants. Further, we also measured the extent in which the participants experience the presence of virtual office walls as useful. The main findings of this study include that virtual office walls contribute to an increased perception on overall performance and that they make work coordination easier.

## 11.3 Evaluation and Threats to Validity

The research conducted as part of this dissertation provides important guidelines on the design and construction of a working environment which constantly adapts itself to provide global software engineers with a sufficient level of awareness. When such a mechanism, which introduces dynamic boundaries instead of static boundaries, is successfully applied to a distributed environment it does not only have the potential to match the co-located setting, it even has the opportunity to outperform it. We studied three important aspects of the design and creation of a virtual office: (i) constructing a virtual office, (ii) communicating in a virtual office, and (iii) information needs in a virtual office. After conducting these studies, we were able to derive a set of requirements a virtual office should fulfill. This set of requirements covers important aspects of the design and creation of a virtual office and should be implemented to support global software engineers in acquiring information awareness related to their current activity.

For each of the empirical studies, we comprehensively described our research sites and methods, made all data-gathering designs available, and discussed the threats to validity. We also made all data available in anonymized form and made the tools we used during the PhD study available upon request. We do this to make both our data gathering process and data analysis process repeatable and to mitigate threats to construct validity.

The research presented in this dissertation, however, also has some limitations. Firstly, it does not provide an all-encompassing approach to support global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness. As such it does not fully reaches the goal of this dissertation, since more work and research is needed to construct a virtual office in which global software engineers can perform their daily activities.

Secondly, we cannot guarantee the completeness of the set of requirements of a virtual office. This is because we only studied three aspects of the design, implementation and evaluation of technological support for aiding global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities. However, it is likely there exists other aspects we did not study. It is also likely that the requirements related to communicating in a virtual office and information needs in a virtual office are not complete. This is because we focused on specific research topics, like overhearing conversations, and did not study the complete research area. Next, the identified requirements should be checked for a number of qualities including correctness, completeness, measurability and consistency.

Thirdly, there exist a threat related to the distributed setting of IHomer. Even though distributed collaboration is in the heart of the company, since the default work location in the company is home, nearly all collaboration takes place within the Netherlands. Therefore, one could argue this does not represents a global software engineering environment since challenges caused by time zone and cultural differences are not encountered.

Fourthly, there exists a minor threat related to the general pattern of studying literature, gathering empirical data from industry, and identifying requirements. This general pattern could have influenced the results, because the author might be biased by literature before starting the data gathering process.

Finally, there exist threats to external validity. In most of the studies presented in this dissertation we only studied a group of participants of a single company, either Exact or IHomer. Therefore, to be able to generalize the findings of these studies, these studies should be repeated in multiple distinct settings. Next to this also the small sample size of most of the studies we conducted makes it difficult to draw statistically significant conclusions.

Chapter 12

# Future Work

In this dissertation we presented our research on how best to support global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities. In the chapters of this dissertation we already provide opportunities for further research. In the remainder of this section we will discuss important next steps of this research and Iris; an implementation of a virtual office, which is used at IHomer.

## 12.1 Recommendations for Future Research

Firstly, it is worth replicating the studies reported in this dissertation in order to draw statistically significant conclusions. Therefore, these studies need to be conducted in other settings as well. Especially the controlled experiment presented in chapter 9 should be replicated. When this replicated experiment results in similar results as the original experiment we can confirm that that virtual office walls contribute to an increased perception on overall performance and that they make work coordination easier. In fact, we can even conclude that our vision on how to provide distributed software engineers with a sufficient level of awareness is correct.

Secondly, the research regarding the requirements of a virtual office should be continued to provide a complete set of requirements. First steps, regarding the requirements related to the information needs of a virtual office are already taken by Kevin Dullemond. In his research he has identified the information types that are important to software engineers, and he investigated how the mutual importance of these items changes as the items are more related to the current activity of a software engineer [Dul13a].

Finally, the work on the identification, implementation, and evaluation of the requirements of a virtual office should be continued. Since, these insights provide valuable information on how best to support global software engineers to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities.

## 12.2 Iris

During our PhD study we have investigated the use of Iris; a support environment in which we implement the requirements of a virtual office, at IHomer. During this study we have developed different versions of Iris each with different focus points and implementation technologies. The version of Iris we discuss in this section is currently used by the majority of the employees at IHomer. In the remainder of this section we briefly describe the process we use to develop this environment, the current state of this environment, and we discuss how this environment fulfills the set of requirements of a virtual office.

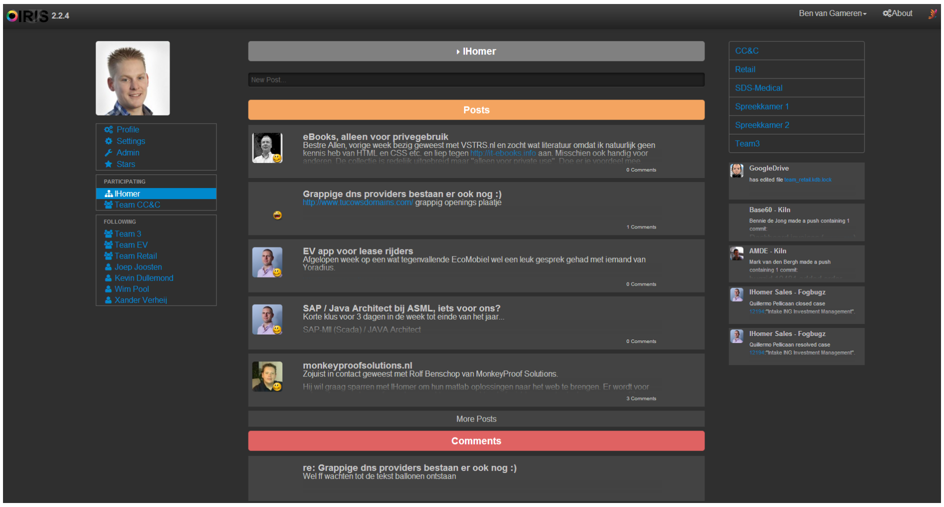
### Development Process

In the development of Iris we use an iterative approach to cope with the uncertainties and changing requirements of developing a genuinely novel product. We also involved the intended users of the system as soon and as strongly as possible. We do this because, on the one hand they are experienced with working in a distributed setting and as such can give valuable feedback. On the other hand we needed to stimulate usage as soon as possible because the value of awareness sharing technology is higher when a larger portion of the team uses it.

### Iris

In figure 12.1 an overview of Iris is shown. The main screen of Iris consists of three sections. Firstly we discuss the left side of the screen. On this side of the screen the user can make changes to his or her personal settings, e.g. the user profile. The user can configure both in which projects and teams he or she participates, and which teams, projects or people he or she follows. Next to configuring their preferences, the users can also select the organization or one of the teams, projects and people to filter the available information. By selecting a specific context the user-interface will only show information that corresponds to that context.

Secondly, in the middle section of the screen the most important information is shown: the posts, comments and events belonging to the selected context. In figure 12.1 the organization context is selected and only the posts and comments which belong to the organization context are visible. In this section users are able to (i) create a new post the selected context and attach an emotion to this, see figure 12.2, (ii) view all posts belonging to this context, (iii) view all comments of all posts belonging to this context, (iv) view detailed information about a post including all comments in chronological order, see figure 12.3, and (v) add a comment to this post.



**eBooks, alleen voor privegebruik**
Bestre Allen, vorige week bezig geweest met VSTRS.nl en zocht wat literatuur omdat ik natuurlijk geen kennis heb van HTML en CSS etc. en liep tegen http://it-ebooks.info aan. Misschien ook handig voor anderen. De collectie is redelijk uitgebreid maar "alleen voor private use". Doe er je voordeel mee.
0 Comments

**Grappige dns providers bestaan er ook nog :)**
http://www.tucowsdomains.com/ grappig openings plaatje
1 Comments

**EV app voor lease rijders**
Afgelopen week op een wat tegenvallende EcoMobiel wel een leuk gesprek gehad met iemand van Yoradius.
0 Comments

**SAP / Java Architect bij ASML, iets voor ons?**
Korte klus voor 3 dagen in de week tot einde van het jaar...
SAP-MII (Scada) / JAVA Architect
0 Comments

**monkeyproofsolutions.nl**
Zojuist in contact geweest met Rolf Benschop van MonkeyProof Solutions.
Hij wil graag sparren met IHomer om hun matlab oplossingen naar het web te brengen. Er wordt voor
3 Comments

**re: Grappige dns providers bestaan er ook nog :)**
Wel ff wachten tot de tekst ballonen ontstaan

**GoogleDrive**
has edited file team_retail.kdb.lock

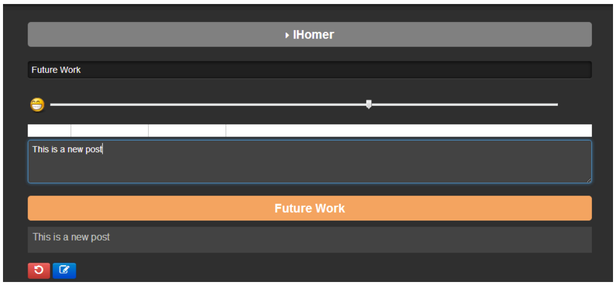
**Base60 - Kiln**
Bennie de Jong made a push containing 1 commit:

**AMDE - Kiln**
Mark van den Bergh made a push containing 1 commit:

**IHomer Sales - Fogbugz**
Quillermo Pellicaan closed case 12194:"Intake ING Investment Management".

**IHomer Sales - Fogbugz**
Quillermo Pellicaan resolved case 12194:"Intake ING Investment Management".

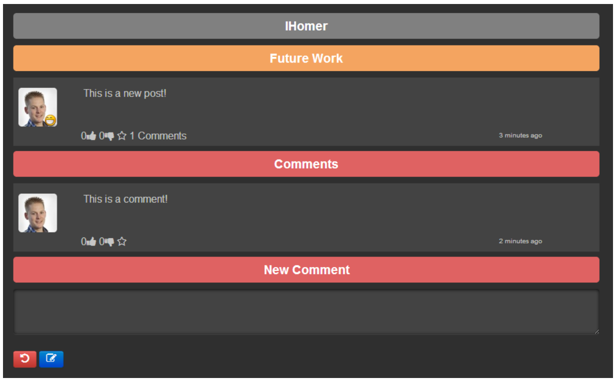
**Figure 12.1:** *Iris - Main Screen*



This is a new post

This is a new post

**Figure 12.2:** *Iris - New Post*



This is a new post!
0 0 1 Comments 3 minutes ago

This is a comment!
0 0 2 minutes ago

**Figure 12.3:** *Iris - Detailed information about a post*

Next to posts and comments, also events can be shown in the middle section of the screen. This can be done by clicking on the summarized event list shown in the bottom right of the screen. By clicking on this list all events belonging to the active context are shown in the middle section of the view. Events in Iris are notifications of actions colleagues are performing in specialized systems that are relevant to software engineers. An example of such an event is reporting a bug in the issue management system. Currently, Iris includes notifications of actions performed in the following specialized systems: (i) Issue Management System, (ii) Software Repository, (iii) Build Sever, (iv) Google Drive, and (v) Google Mail. Changes to issues originating from the issue management system are shown in Iris. Users get informed of the creation of a new issue, the assignment of an issue to someone, and when an issue is resolved. Next, the pushes to the software repository are also shown in Iris. Furthermore, also the results of the automated builds and deployments are shown to the users. Users get also informed of changes to files in the Google Drive folder of the company, notifying the users something has been edited. Finally, also changes to files in the Google Drive folders of the company are shown as events. Finally, it is also possible to forward mails to Iris which will automatically be shared in the specified context.

Finally, in the right section of the screen not only a summarized list of events is shown, but also a list of links to various Google Hangouts the user can use to communicate. Each of this links has an indicator showing who are currently present in the Hangout.

## A Virtual Office

Finally, we discuss how Iris fulfills the set of requirements of a virtual office. The results are combined in table 12.1. In this table a *'✗-sign'* indicates that the requirement depicted in the row is not implemented by the solution depicted in the column. A *'✓-sign'* indicates that progress in fulfilling this requirement is made.

First, we discuss the requirements related to constructing a virtual office. Because we develop Iris based on our vision on how best to provide distributed software engineers with a sufficient level of awareness for their work activities, it attempts to fulfill the first five requirements. As described in the previous section Iris already combines information of actions performed in multiple specialized systems (Req 3.) and provides a mechanism, based on the current context of a software engineer, to differentiate between required and not required information (Req 4. and Req 5.). It also facilitates having access to a data set which contains important information about the actions colleagues performed in specialized tooling. This can however be improved by including other relevant systems (Req 2.). Finally, it is to early to conclude that by using the Iris platform engineers are able to acquire awareness relatively passively and unobtrusively (Req 1.). We do however have initial results that our approach is valuable [Dul13c].

Secondly, we discuss the requirements related to communicating in a virtual office. Employees at IHomer use Google Hangouts to have conversations. As such we did not implement this functionality in Iris. Hence, we discuss how Google Hangouts fulfill the requirements related to having and overhearing conversations. In a Google Hangout it is possible to both have conversations with anyone who is present, and to overhear conversations others are having. As such Google Hangouts fulfill requirement six to ten. In these Hangouts it is also possible for users to change their degree of involvement in a conversation, they could for example join a conversation which they overhear. However, it is not possible to have private conversations (Req 11.), and to find out about finished conversations (Req 13.). Next to having and overhearing conversations, users should also be able to share interesting information and express their current mood. In the previous section we discussed that it is possible to create new posts and add comments to these posts. In Iris it is also possible for users to express their mood about what they are posting, see figure 12.2. As such requirement 14 to 16 are fulfilled by Iris.

Finally, we discuss the requirements related to information needs of a virtual office. Currently, Iris partially fulfills requirement 17 and 18. Since most of the coupled specialized systems contain information about the technological solution, e.g. issue management system and the software repository. Other coupled systems contain information about the completed artifacts, e.g. Google Drive and the build server. These systems do not include all information needed by the software engineers and as such other specialized systems should be coupled. The current solution of Iris does not have a mechanism to interrupt software engineers to provide them with new important information (Req 19. and Req 20.). Therefore we should design such a mechanism and evaluate it in an industrial setting.

It is exiting to continue the design and construction of a virtual office, which automatically regulates the available information based on both the current activity of an engineer, and the information engineers want to know immediately.

| **Requirement** | | **Iris** | **Hangouts** |
|---|---|---|---|
| Req 1. | Facilitate acquiring awareness relatively passively and unobtrusively | ✓ | ✗ |
| Req 2. | Facilitate having access to a data set which at least contains the required data to carry out work activities | ✓ | ✗ |
| Req 3. | Facilitate combining data from different sources | ✓ | ✗ |
| Req 4. | Facilitate differentiating between required and not required information | ✓ | ✗ |
| Req 5. | Facilitate a valid representation of the context of a software engineer | ✓ | ✗ |
| Req 6. | Facilitate starting conversations | ✗ | ✓ |
| Req 7. | Facilitate detecting active conversations | ✗ | ✓ |
| Req 8. | Facilitate monitoring active conversations | ✗ | ✓ |
| Req 9. | Facilitate participating in conversations | ✗ | ✓ |
| Req 10. | Facilitate finishing conversations | ✗ | ✓ |
| Req 11. | Facilitate having conversations which cannot be overheard by others | ✗ | ✗ |
| Req 12. | Facilitate changing the degree of involvement in a conversation | ✗ | ✓ |
| Req 13. | Facilitate having insight in the finished conversations | ✗ | ✗ |
| Req 14. | Facilitate sharing information about a new topic | ✓ | ✗ |
| Req 15. | Facilitate responding to an existing topic | ✓ | ✗ |
| Req 16. | Facilitate sharing mood | ✓ | ✗ |
| Req 17. | Facilitate informing software engineers immediately of information about completed artifacts | ✓ | ✗ |
| Req 18. | Facilitate informing software engineers immediately of information about the technological solution | ✓ | ✗ |
| Req 19. | Facilitate interrupting software engineers to provide them with new information | ✗ | ✗ |
| Req 20. | Facilitate controlling the moments during which one prefers not to be interrupted | ✗ | ✗ |

**Table 12.1:** *How Iris fulfills the requirements of a Virtual Office*

# Bibliography

[Abr02] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta. *Agile software development methods. Review and analysis.* VTT Publications, 2002.

[Åge05] P. J. Ågerfalk, B. Fitzgerald, H. Holmström, B. Lings, B. Lundell, and E. Ó. Conchúir. A framework for considering opportunities and threats in distributed software development. In *Proceedings of the International Workshop on Distributed Software Development*, pp. 47–61. Austrian Computer Society, 2005.

[Åge08] P. J. Ågerfalk, B. Fitzgerald, H. H. Olsson, and E. Ó. Conchúir. Benefits of global software development: The known and unknown. In *Making Globally Distributed Software Development a Success Story*, pp. 1–9. Springer, 2008.

[All77] T. Allen. *Managing the flow of technology.* MIT press, 1977.

[All07] T. Allen and G. Henn. *The organization and architecture of innovation: Managing the flow of technology.* Elsevier, 2007.

[Bab12] E. R. Babbie. *The Practice of Social Research.* Cengage Learning, 13 edn., 2012.

[Bai78] K. D. Bailey. *Methods of Social Research.* Free Press, 1978.

[Bat01] R. D. Battin, R. Crocker, J. Kreidler, and K. Subramanian. Leveraging resources in global software development. *Software*, vol. 18(2):pp. 70–77, 2001.

[Bir08] J. Birnholtz, C. Gutwin, G. Ramos, and M. Watson. OpenMessenger: gradual initiation of interaction for distributed workgroups. In *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 1661–1664. ACM, 2008.

[Bly93] S. A. Bly, S. R. Harrison, and S. Irwin. Media spaces: bringing people together in a video, audio, and computing environment. *Communications*, vol. 36(1):pp. 28–46, 1993.

[Cad01] J. Cadiz, G. Venolia, G. Jancke, and A. Gupta. Sideshow: Providing peripheral awareness of important information. Tech. rep., Microsoft Research, Collaboration, and Multimedia Group, 2001.

[Cal12] F. Calefato, F. Lanubile, T. Conte, and R. Prikladnicki. Assessing the impact of real-time machine translation on requirements meetings: a replicated experiment. In *Proceedings of the international symposium on Empirical software engineering and measurement*, pp. 251–260. ACM, 2012.

[Car99] E. Carmel. *Global software teams: collaborating across borders and time zones.* Prentice Hall PTR, 1999.

[Car01] E. Carmel and R. Agarwal. Tactical approaches for alleviating distance in global software development. *Software*, vol. 18(2):pp. 22–29, 2001.

[Cat06] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In *Proceedings of the conference on Computer Supported Cooperative Work*, pp. 353–362. ACM, 2006.

[Con06] E. Ó. Conchúir, H. H. Olsson, P. J. Ågerfalk, and B. Fitzgerald. Exploring the assumed benefits of global software development. In *Proceedings of the International Conference on Global Software Engineering*, pp. 159–168. IEEE, 2006.

[Cro05] K. Crowston and J. Howison. The social structure of free and open source software development. *First Monday*, vol. 10(2), 2005.

[Cur88] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, vol. 31(11):pp. 1268–1287, 1988.

[Cut01] E. Cutrell, M. Czerwinski, and E. Horvitz. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. pp. 263–269. IOS, 2001.

[Cze04] M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proceedings of the conference on Human factors in Computing Systems*, pp. 175–182. ACM, 2004.

[Dab11] L. Dabbish, G. Mark, and V. M. González. Why do i keep interrupting myself?: environment, habit and self-interruption. In *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 3127–3130. ACM, 2011.

[Dal63] N. Dalkey and O. Helmer. An experimental application of the delphi method to the use of experts. *Management science*, vol. 9(3):pp. 458–467, 1963.

[Dam06] D. Damian and D. Moitra. Global software development: How far have we come? *Software*, vol. 23(5):pp. 17–19, 2006.

[Die09] Dieringer Research Group Inc. Telework Trendlines 2009: A Survey Brief by WorldatWork, 2009.

[Dix04] A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-computer interaction.* Prentice hall, 2004.

[Dou92] P. Dourish and V. Bellotti. Awareness and Coordination in Shared Workspaces. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 107–114. ACM, 1992.

[DS11] C. R. De Souza and D. F. Redmiles. The awareness network, to whom should i display my actions? and, whose actions should i monitor? *Transactions on Software Engineering*, vol. 37(3):pp. 325–340, 2011.

[Dul09] K. Dullemond and B. van Gameren. *Technological support for distributed agile development.* Master thesis, Delft University of Technology, 2009.

[Dul10] K. Dullemond, B. van Gameren, and R. van Solingen. Virtual open conversation spaces: Towards improved awareness in a gse setting. In *Proceedings of the 5th International Conference on Global Software Engineering*, pp. 247–256. IEEE, 2010.

[Dul11a] K. Dullemond and B. van Gameren. Communico: overhearing conversations in a virtual office. In *Proceedings of the conference on Computer Supported Cooperative Work*, pp. 577–578. ACM, 2011.

[Dul11b] K. Dullemond, B. van Gameren, and R. van Solingen. An exploratory study on open conversation spaces in software engineering. In *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 307–316. IEEE, 2011.

[Dul11c] K. Dullemond, B. van Gameren, and R. van Solingen. Overhearing conversations in global software engineering-requirements and an implementation. In *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1–8. IEEE, 2011.

[Dul12a] K. Dullemond and B. van Gameren. An industrial evaluation of technological support for overhearing conversations in global software engineering. In *Proceedings of the 7th International Conference on Global Software Engineering*, pp. 65–74. IEEE, 2012.

[Dul12b] K. Dullemond, B. van Gameren, and R. van Solingen. Supporting distributed software engineering in a fully distributed organization. In *Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 30–36. IEEE, 2012.

[Dul13a] K. Dullemond and B. van Gameren. What distributed software teams need to know and when: an empirical study. In *Proceedings of the 8th International Conference on Global Software Engineering*, pp. 61–70. IEEE, 2013.

[Dul13b] K. Dullemond, B. v. van Gameren, M.-A. Storey, and A. v. Deursen. Fixing the'out of sight out of mind'problem: one year of mood-based microblogging in a distributed software team. In *Proceedings of the 10th International Workshop on Mining Software Repositories*, pp. 267–276. IEEE, 2013.

[Dul13c] K. Dullemond and R. van Solingen. Increasing awareness in distributed software teams: a first evaluation. In *Proceedings of the 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 325–334. IEEE, 2013.

[Dun64] M. D. Dunnette. *Are meetings any good for solving problems?* Selbstverl., 1964.

[Ebe01] C. Ebert and P. De Neve. Surviving global software development. *Software*, vol. 18(2):pp. 62–69, 2001.

[Ehr07] K. Ehrlich, G. Valetto, and M. Helander. Seeing inside: Using social network analysis to understand patterns of collaboration and coordination in global software teams. In *Proceedings of the International Conference on Global Software Engineering*, pp. 297–298. IEEE, 2007.

[Ehr10] K. Ehrlich and N. S. Shami. Microblogging inside and outside the workplace. In *Proceedings of the international conference on Supporting group work*, pp. 42–49. AAAI, 2010.

[Eng87] Y. Engeström. *Learning by expanding. An activity-theoretical approach to developmental research.* Orienta-Konsultit Oy, 1987.

[Eng99] Y. Engeström and R. Miettinen. *Perspectives on activity theory.* Cambridge University, 1999.

[Epp04] M. J. Eppler and J. Mengis. The concept of information overload: A review of literature from organization science, accounting, marketing, mis, and related disciplines. *The information society*, vol. 20(5):pp. 325–344, 2004.

[Eri99] T. Erickson, D. N. Smith, W. A. Kellogg, M. Laff, J. T. Richards, and E. Bradner. Socially translucent systems: social proxies, persistent conversation, and the design of babble. In *Proceedings of the conference on Human Factors in Computing Systems*, pp. 72–79. ACM, 1999.

[Eri06] T. Erickson, W. Kellogg, M. Laff, J. Sussman, T. Wolf, C. Halverson, and D. Edwards. A persistent chat space for work groups: the design, evaluation and deployment of loops. In *Proceedings of the Conference on Designing Interactive systems*, pp. 331–340. ACM, 2006.

[Esp03] J. A. Espinosa and E. Carmel. The impact of time separation on coordination in global software teams: a conceptual foundation. *Software Process: Improvement and Practice*, vol. 8(4):pp. 249–266, 2003.

[Esp07] J. A. Espinosa, N. Nan, and E. Carmel. Do gradations of time zone separation make a difference in performance? a first laboratory study. In *Proceedings of the International Conference on Global Software Engineering*, pp. 12–22. IEEE, 2007.

[Exa10] Exact Holding N.V. Annual Report 2010, 2010.

[Fes12] A. Fessl, V. Rivera-Pelayo, V. Pammer, and S. Braun. Mood tracking in virtual meetings. In *Proceedings of the European Conference of Technology Enhanced Learning*, pp. 377–382. Springer, 2012.

[Fin03] A. Fink. *How to manage, analyze, and interpret survey data.* Sage, London, 2nd edn., 2003.

[Fis35] R. Fisher. *The design of experiments.* Oliver & Boyd, 1935.

[Fis12] G. Fischer. Context-aware systems: the 'right' information, at the 'right' time, in the 'right' place, in the 'right' way, to the 'right' person. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pp. 287–294. ACM, 2012.

[Fog05] J. Fogarty, S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. C. Lee, and J. Yang. Predicting human interruptibility with sensors. *Transactions on Computer-Human Interaction*, vol. 12(1):pp. 119–146, 2005.

[Fon05] A. Fontana and J. Frey. The interview: From neutral stance to political involvement. *The Sage handbook of qualitative research*, vol. 3:pp. 695–727, 2005.

[Ful13] D. Fullerton. Why we (still) believe in working remotely, 2013.

[Gal90] J. E. Galegher, R. E. Kraut, and C. E. Egido. *Intellectual teamwork: Social and technological foundations of cooperative work.* Lawrence Erlbaum Associates, 1990.

[Gam12] B. van Gameren, K. Dullemond, and R. van Solingen. Auto-erecting virtual office walls. In *Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 391–397. IEEE, 2012.

[Gam13a] B. van Gameren, R. van Solingen, and K. Dullemond. Auto-erecting virtual office walls a controlled experiment. In *Proceedings of the 8th International Conference on Global Software Engineering*, pp. 206–215. IEEE, 2013.

[Gam13b] B. v. van Gameren and R. van Solingen. When to interrupt global software engineers to provide them with what information? In *Proceedings of the 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 495–504. IEEE, 2013.

[Gar99] O. García, J. Favela, and R. Machorro. Emotional awareness in collaborative systems. In *String Processing and Information Retrieval Symposium, 1999 and International Workshop on Groupware*, pp. 296–303. IEEE, 1999.

[Gib97] A. Gibbs. Focus groups. *Social research update*, vol. 19(8), 1997.

[Gil00] B. Gillham. *Developing a questionnaire*. Continuum, 2000.

[Gre01] S. Greenberg and M. Rounding. The notification collage: posting information to public and personal displays. In *Proceedings of the conference on Human factors in Computing Systems*, pp. 514–521. ACM, 2001.

[Gre02] J. Grenning. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, 2002.

[Gre07] K. C. Green, J. S. Armstrong, and A. Graefe. Methods to elicit forecasts from groups: Delphi and prediction markets compared. *Foresight: The International Journal of Applied Forecasting*, vol. 8(8):pp. 17–21, 2007.

[Gri99] R. E. Grinter, J. D. Herbsleb, and D. E. Perry. The geography of coordination: dealing with distance in r&d work. In *Proceedings of the International Conference on Supporting Group Work*, pp. 306–315. ACM, 1999.

[Gro05] T. Gross, C. Stary, and A. Totter. User-centered awareness in computer-supported cooperative work-systems: Structured embedding of findings from social sciences. *International Journal of Human-Computer Interaction*, vol. 18(3):pp. 323–360, 2005.

[Gus73] D. H. Gustafson, R. K. Shukla, A. Delbecq, and G. W. Walster. A comparative study of differences in subjective likelihood estimates made by individuals, interacting groups, delphi groups, and nominal groups. *Organizational Behavior and Human Performance*, vol. 9(2):pp. 280–291, 1973.

[Gut95] C. Gutwin, G. Stark, and S. Greenberg. Support for workspace awareness in educational groupware. In *The International Conference on Computer Support for Collaborative Learning*, pp. 147–156. L. Erlbaum Associates Inc., 1995.

[Gut96] C. Gutwin, S. Greenberg, and M. Roseman. Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. In *People and Computers*, pp. 281–298. Springer, 1996.

[Gut02] C. Gutwin and S. Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work*, vol. 11(3-4):pp. 411–446, 2002.

[Gut04] C. Gutwin, R. Penner, and K. Schneider. Group awareness in distributed software development. In *Proceedings of the conference on Computer Supported Cooperative Work*, pp. 72–81. ACM, 2004.

[Har02] I. Harpaz. Advantages and disadvantages of telecommuting for the individual, organization and society. *Work Study*, vol. 51(2):pp. 74–80, 2002.

[Her99] J. D. Herbsleb and R. E. Grinter. Architectures, coordination, and distance: Conway's law and beyond. *Software*, vol. 16(5):pp. 63–70, 1999.

[Her00] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter. Distance, dependencies, and delay in a global collaboration. In *Proceedings of the conference on Computer Supported Cooperative Work*, pp. 319–328. ACM, 2000.

[Her01] J. D. Herbsleb and D. Moitra. Global software development. *Software*, vol. 18(2):pp. 16–20, 2001.

[Her03] J. D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. *Transactions on Software Engineering*, vol. 29(6):pp. 481–494, 2003.

[Her05] J. D. Herbsleb, D. J. Paulish, and M. Bass. Global software development at siemens: experience from nine projects. In *Proceedings of the International Conference on Software Engineering*, pp. 524–533. IEEE, 2005.

[Her07] J. D. Herbsleb. Global software engineering: The future of socio-technical coordination. In *Future of Software Engineering*, pp. 188–198. IEEE, 2007.

[Hes91] B. W. Hesse and C. E. Grantham. Electronically distributed work communities: implications for research on telework. *Internet Research*, vol. 1(1):pp. 4–17, 1991.

[Hil85] S. R. Hiltz and M. Turoff. Structuring computer-mediated communication systems to avoid information overload. *Communications*, vol. 28(7):pp. 680–689, 1985.

[Hol06] H. Holmström, E. Ó. Conchúir, P. J. Ågerfalk, and B. Fitzgerald. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In *Proceedings of the International Conference on Global Software Engineering*, pp. 3–11. IEEE, 2006.

[Ink08] K. Inkpen, S. Whittaker, M. Czerwinski, R. Fernandez, and J. Wallace. GroupBanter: Supporting Serendipitous Group Conversations with IM. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 485–498. Springer, 2008.

[Iqb07] S. T. Iqbal and E. Horvitz. Disruption and recovery of computing tasks: field study, analysis, and directions. In *Proceedings of the conference on Human factors in Computing Systems*, pp. 677–686. ACM, 2007.

[Iqb10] S. T. Iqbal and E. Horvitz. Notifications and awareness: a field study of alert usage and preferences. In *Proceedings of the conference on Computer Supported Cooperative Work*, pp. 27–30. ACM, 2010.

[Jan08] B. Jansen, I. Taksa, and A. Spink. Research and methodological foundations of transaction log analysis. In *Handbook of research on Web log analysis*, pp. 1–17. Hershey, 2008.

[Jav07] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the Workshop on Web Mining and Social Network Analysis*, pp. 56–65. ACM, 2007.

[Jud91] C. M. Judd, E. R. Smith, and L. H. Kidder. *Research Methods in Social Relations.* Harcourt Brace Jovanovich, 1991.

[Kel03] K. Kelley, B. Clark, V. Brown, and J. Sitzia. Good practice in the conduct and reporting of survey research. *International Journal for Quality in Health Care*, vol. 15(3):pp. 261–266, 2003.

[Kie03] L. Kiel. Experiences in distributed development: a case study. In *Proceedings of the International Workshop on Global Software Development*, pp. 44–47. 2003.

[Ko07] A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *Proceedings of the International Conference on Software Engineering*, pp. 344–353. IEEE, 2007.

[Kon04] J. Kontio, L. Lehtola, and J. Bragge. Using the focus group method in software engineering: obtaining practitioner and user experiences. In *Proceedings of the International Symposium on Empirical Software Engineering*, pp. 271–280. IEEE, 2004.

[Kra90] R. E. Kraut, R. S. Fish, R. W. Root, and B. L. Chalfonte. Informal communication in organizations: Form, function, and technology. In *Human reactions to technology: Claremont symposium on applied social psychology*, pp. 145–199. Sage Publications, 1990.

[Kra95] R. E. Kraut and L. A. Streeter. Coordination in software development. *Communications*, vol. 38(3):pp. 69–81, 1995.

[Lan03] J. D. Langford and D. McDonagh. *Focus Groups: Supporting Effective Product Development.* Taylor and Francis, 2003.

[Lan10] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno. Collaboration tools for global software engineering. *Software*, vol. 27(2):pp. 52–55, 2010.

[Lar04] C. Larman. *Agile and iterative development: a manager's guide.* Addison-Wesley Professional, 2004.

[Leo78] A. Leont'ev. *Activity, Consciousness, and Personality.* Englewood Cliffs, Prentice Hall, 1978.

[Lik32] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, vol. 22(140):pp. 1–55, 1932.

[Mar02] G. Mark. Extreme collaboration. *Communications of the ACM*, vol. 45(6):pp. 89–93, 2002.

[Mar08] G. Mark, D. Gudith, and U. Klocke. The cost of interrupted work: more speed and stress. In *Proceedings of the conference on Human Factors in Computing Systems*, pp. 107–110. ACM, 2008.

[McC93] K. R. McCord. *Managing the integration problem in concurrent engineering.* Master thesis, Massachusetts Institute of Technology, 1993.

[Mis03] F. Mish, editor. *Merriam-Webster's collegiate dictionary.* Merriam-Webster Inc., Springfield, 2003.

[Mor11] S. Mora, V. Rivera-Pelayo, and L. Müller. Supporting mood awareness in collaborative settings. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 268–277. IEEE, 2011.

[Ngu08] T. Nguyen, T. Wolf, and D. Damian. Global software development and delay: Does distance still matter? In *Proceedings of the International Conference on Global Software Engineering*, pp. 45–54. IEEE, 2008.

[Nii08] T. Niinimaki and C. Lassenius. Experiences of instant messaging in global software development projects: A multiple case study. In *Proceedings of the International Conference on Global Software Engineering*, pp. 55–64. IEEE, 2008.

[Oko04] C. Okoli and S. D. Pawlowski. The delphi method as a research tool: an example, design considerations and applications. *Information & Management*, vol. 42(1):pp. 15–29, 2004.

[Ols96] J. S. Olson and S. Teasley. Groupware in the wild: lessons learned from a year of virtual collocation. In *Proceedings of the conference on Computer Supported Cooperative Work*, pp. 419–427. ACM, 1996.

[Ols00] G. M. Olson and J. S. Olson. Distance matters. *Human-computer interaction*, vol. 15(2):pp. 139–178, 2000.

[Omo09] I. Omoronyia, J. Ferguson, M. Roper, and M. Wood. Using developer activity data to enhance awareness during collaborative software development. *Computer Supported Cooperative Work*, vol. 18:pp. 509–558, 2009.

[Omo10] I. Omoronyia, J. Ferguson, M. Roper, and M. Wood. A review of awareness in distributed collaborative software engineering. *Software: Practice and Experience*, vol. 40(12):pp. 1107–1133, 2010.

[O'R04] T. O'Reilly. The architecture of participation, 2004.

[Par76] D. Parnas. On the design and development of program families. *Transactions on Software Engineering*, vol. SE-2(1):pp. 1–9, 1976.

[Per94] D. E. Perry, N. A. Staudenmayer, and L. G. Votta. People, organizations, and process improvement. *Software*, vol. 11(4):pp. 36–45, 1994.

[Pot93] C. Potts. Software-engineering research revisited. *Software, IEEE*, vol. 10(5):pp. 19–28, 1993.

[PR12] J. Portillo-Rodríguez, A. Vizcaíno, M. Piattini, and S. Beecham. Tools used in global software engineering: A systematic mapping review. *Information and Software Technology*, vol. 54(7):pp. 663 – 685, 2012.

[Pra93] J. Pratt. Myths and Realities of Working at Home: Characteristics of Homebased Business Owners and Telecommuters. Tech. rep., National Technical Information Service, 1993.

[Pri99] W. Prinz. Nessie: an awareness environment for cooperative settings. In *Proceedings of the European Conference on Computer Supported Cooperative Work*, pp. 391–410. Springer, 1999.

[Pri07] R. Prikladnicki, J. L. N. Audy, D. Damian, and T. C. de Oliveira. Distributed software development: Practices and challenges in different business strategies of offshoring and onshoring. In *Proceedings of the International Conference on Global Software Engineering*, pp. 262–274. IEEE, 2007.

[Pri09] R. Prikladnicki. Exploring propinquity in global software engineering. In *Proceedings of the International Conference on Global Software Engineering*, pp. 133–142. IEEE, 2009.

[Red07] D. Redmiles, A. Van Der Hoek, B. Al-Ani, T. Hildenbrand, S. Quirk, A. Sarma, R. Filho, C. de Souza, and E. Trainer. Continuous coordination-a new paradigm to support globally distributed software development projects. *Wirtschafts Informatik*, vol. 49(1):p. 28, 2007.

[Ren11] Y. Ren and R. E. Kraut. A simulation for designing online community: Member motivation, contribution, and discussion moderation, 2011.

[Rib02] A. Ribak, M. Jacovi, and V. Soroka. Ask before you search: peer support and community building with ReachOut. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 126–135. ACM, 2002.

[San06] R. Sangwan, M. Bass, N. Mullick, D. J. Paulish, and J. Kazmeier. *Global software development handbook*. Auerbach Publications, 2006.

[Sar05] A. Sarma. A survey of collaborative tools in software development. Tech. rep., University of California, Irvine, 2005.

[Sar09] A. Sarma, J. Herbsleb, L. Maccherone, and P. Wagstrom. Tesseract: Interactive visual exploration of socio-technical relationships in software development. In *Proceedings of the International Conference on Software Engineering*, pp. 23–33. IEEE, 2009.

[Sar10] A. Sarma, A. Van der Hoek, and D. Redmiles. The coordination pyramid: A perspective on the state of the art in coordination technology. *Computer*, vol. PP(99):p. 1, 2010.

[Sch95] K. Schwaber. Scrum development process. In *Proceedings of the Conference on Object Oriented Programming Systems, Languages, and Applications*, pp. 117–134. Springer, 1995.

[Sch02] K. Schmidt. The Problem with 'Awareness': Introductory Remarks on 'Awareness in CSCW'. *Computer Supported Cooperative Work*, vol. 11(3-4):pp. 285 – 298, 2002.

[Sch11] K. Schwaber and J. Sutherland. Scrum guide. *Scrum Alliance*, 2011.

[Seg95] L. Segal. Designing team workstations: The choreography of teamwork. *Local applications of the ecological approach to human-machine systems*, vol. 2, 1995.

[Sha11] G. Sharma, G. Shroff, and P. Dewan. Workplace collaboration in a 3d virtual office. In *Proceedings of the International Symposium on VR Innovation*, pp. 3–10. IEEE, 2011.

[Sil08] J. Sillito, G. C. Murphy, and K. De Volder. Asking and answering questions during a programming change task. *Transactions on Software Engineering*, vol. 34(4):pp. 434–451, 2008.

[Sim96] H. A. Simon. *The sciences of the artificial.* MIT press, 1996.

[Smi00] M. Smith, J. Cadiz, and B. Burkhalter. Conversation trees and threaded chats. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 97–105. ACM, 2000.

[Soa03] C. Soanes, editor. *The Oxford compact English dictionary.* Oxford University Press, New York, 2003.

[Sol98] R. van Solingen, E. Berghout, and F. van Latum. Interrupts: just a minute never is. *Software*, vol. 15(5):pp. 97–103, 1998.

[Sol10] R. van Solingen and M. Valkema. The impact of number of sites in a follow the sun setting on the actual and perceived working speed and accuracy: A controlled experiment. In *Proceedings of the International Conference on Global Software Engineering*, pp. 165–174. IEEE, 2010.

[Sos02] M. E. Sosa, S. D. Eppinger, M. Pich, D. G. McKendrick, and S. K. Stout. Factors that influence technical communication in distributed product development: an empirical study in the telecommunications industry. *Transactions on Engineering Management*, vol. 49(1):pp. 45–58, 2002.

[Sto05] M.-A. D. Storey, D. Čubranić, and D. M. German. On the use of visualization to support awareness of human activities in software development: a survey and a framework. In *Proceedings of the Symposium on Software visualization*, pp. 193–202. ACM, 2005.

[Sto10] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng. The impact of social media on software engineering practices and tools. In *Proceedings of the Workshop on Future of software engineering research*, pp. 359–364. ACM, 2010.

[Syr97] A. Syri. Tailoring cooperation support through mediators. In *Proceedings of the European Conference on Computer Supported Cooperative Work*, pp. 157–172. Springer, 1997.

[Tea10] C. P. Team. Cmmi for development, version 1.3, improving processes for developing better products and services. *no CMU/SEI-2010-TR-033 Software Engineering Institute*, 2010.

[Tel12] P. Tell and M. A. Babar. Activity theory applied to global software engineering: Theoretical foundations and implications for tool builders. In *Proceedings of the International Conference on Global Software Engineering*, pp. 21–30. IEEE, 2012.

[Tra05] M. Tran, Y. Yang, and G. Raikundalia. Supporting awareness in instant messaging: an empirical study and mechanism design. In *Proceedings of the Australian Conference on Computer Human Interaction: Citizens Online: Considerations for Today and the Future*, pp. 1–10. Computer-Human Interaction Special Interest Group of Australia, 2005.

[Tra08] G. Trapani and A. Pash. The complete guide to google wave, 2008.

[Ven10] G. Venolia, J. Tang, R. Cervantes, S. Bly, G. Robertson, B. Lee, and K. Inkpen. Embodied social proxy: mediating interpersonal connection in hub-and-satellite teams. In *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 1049–1058. ACM, 2010.

[Vro69] V. H. Vroom, L. D. Grant, and T. S. Cotton. The consequences of social interaction in group problem solving. *Organizational Behavior and Human Performance*, vol. 4(1):pp. 77–95, 1969.

[Wan13] Y. Wang and D. Redmiles. Understanding cheap talk and the emergence of trust in global software engineering: An evolutionary game theory perspective. In *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 149–152. IEEE, 2013.

[Web93] A. M. Webber. What's so new about the new economy? *Harvard Business Review*, vol. 71:pp. 24–24, 1993.

[Woh00] C. Wohlin, P. Runeson, M. Host, C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction.* Kluver Academic, 2000.

[Woo08] K. Woodford, editor. *Cambridge advanced learner's dictionary.* Cambridge University Press, Cambridge, 2008.

[Wyn79] E. H. Wynn. *Office conversation as an information medium.* Department of Anthropology, University of California, 1979.

[Zha09] D. Zhao and M. B. Rosson. How and why people twitter: the role that micro-blogging plays in informal communication at work. In *Proceedings of the International Conference on Weblogs and Social Media.* ACM, 2009.

[Zha10] J. Zhang, Y. Qu, J. Cody, and Y. Wu. A case study of micro-blogging in the enterprise: use, value, and related issues. In *Proceedings of the international conference on Human factors in Computing Systems*, pp. 123–132. ACM, 2010.

# Appendix A

# Open Conversation Spaces - Discussion Guide

*This appendix depicts the document used to structure the focus group to acquire the empirical data to determine (i) benefits and challenges of having insight in active conversations, (ii) important types of information about a conversation, (iii) actions that are possible with respect to a conversation, and (iv) benefits and challenges of having insight in finished conversations. This focus group is discussed in chapter 5.*

## Focus Group Discussion Guide

### Objectives and information needs

To evaluate the Virtual Open Conversation Space paradigm in general.

### Themes

- Overhearing Conversations
- Conversations
- Finished Conversations
- General

### Overhearing Conversations

What do you think are the benefits of overhearing conversations of others?

What do you think are the challenges of overhearing conversations of others?

What do you think are the benefits of being overheard by others?

What do you think are the challenges of being overheard by others?

### Conversations

What do you think is particularly important about conversations themselves?
(Information Items/Actions)

### Finished Conversations

What do you think are the benefits of seeing the past conversations of your colleagues?

What do you think are the challenges of seeing the past conversations of your colleagues?

What do you think are the benefits of seeing the past conversations of your own?

What do you think are the challenges of seeing the past conversations of your own?

### General

Do you have other things you would like to mention?
(Benefits/Challenges/Ideas/ etc.)

Appendix B

# Open Conversation Spaces - Survey

*This appendix depicts the document used to determine the relative importance of the (i) benefits and challenges of having insight in active conversations, (ii) information items about a conversation, (iii) actions that are possible with respect to a conversation, and (iv) benefits and challenges of having insight in finished conversations. This survey is discussed in chapter 5.*

## Communico Survey

## Communico at Exact

Thanks for using Communico. We hope you take the time to fill in this survey as it is very important to us and our research (note: most open questions are optional). The goal of this survey is to evaluate the Virtual Open Conversation space paradigm and the use of Communico within Exact in general. An Open Conversation space is a space in which it is possible to both have and overhear conversations. A well-known example is the traditional office setting and with Communico we have attempted to create one that is applicable in a distributed setting as well.

The survey was created based on data gathered from a focus group held within Exact with 8 users of Communico and interviews with 4 people working physically dislocated form the Exact office in Delft. In the survey it is made clear which questions to answer when you have not used Communico and which to answer when you have. Both these viewpoints are important to us. You are free to answer the questions in this survey in Dutch or in English. Your answers will be kept confidential. Thank you for your participation.

### Personal Information

**Providing the following information is optional.**

First Name: ______________________ Last Name: ______________________________
Address: _______________________________________________________________
City: ___________________________ ZIP Code: ______________
Telephone: ______________________ Gender: _______________ Age: _____________

### General Information

**G1 What is your current function within Exact?**

_______________________________________________________________________

**G2 How many years have you worked in your current field of expertise (i.e. Software Engineering)?**

_______________________________________________________________________

**G3 Do you work physically dislocated from some of your direct colleagues?**

○ Yes ○ No

**G4 If so, please explain the situation. For example how many of your direct colleagues are dislocated from you and how often is this the case?**

_______________________________________________________________________
_______________________________________________________________________

**Communico Survey**

**G5 How often have you used Communico?**

| ○ | ○ | ○ | ○ | ○ |
|---|---|---|---|---|
| Never | Rarely | About one week | Several weeks | Several months |

**G6 Why did you either use or not use Communico?**

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

**G7 When you used Communico, how many hours a day would you generally use it?**

____________________________________________________________

____________________________________________________________

## Conceptual Questions

Now we turn to the concept of an Open Conversation Space. In such a space (i.e. traditional office setting) people are able to both have and overhear conversations of each other.

### Overhearing Conversations

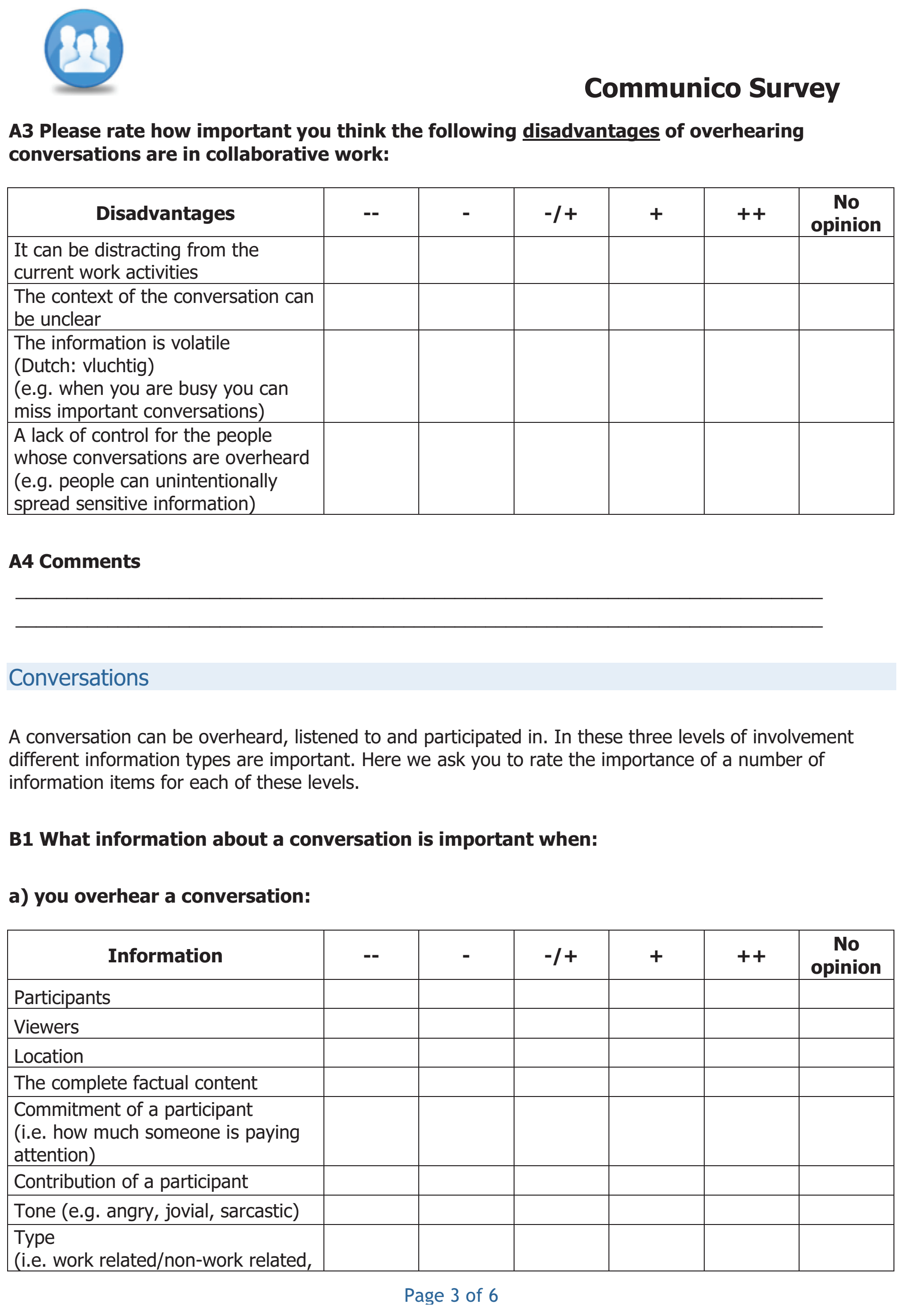
**A1 Please rate how important you think the following advantages of overhearing conversations are in collaborative work:**

| **Advantages** | **--** | **-** | **-/+** | **+** | **++** | **No opinion** |
|---|---|---|---|---|---|---|
| Access to the technical knowledge of colleagues | | | | | | |
| Acquiring involvement (Dutch: betrokkenheid) with your colleagues | | | | | | |
| Enjoying your work (e.g. overhearing a joke) | | | | | | |
| Acquiring insight in the communication structure (e.g. If someone you are looking for is absent you can contact someone he often speaks with) | | | | | | |
| Being able to join a conversation | | | | | | |

**A2 Comments**

____________________________________________________________

____________________________________________________________



## Communico Survey

**A3 Please rate how important you think the following <u>disadvantages</u> of overhearing conversations are in collaborative work:**

| Disadvantages | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| It can be distracting from the current work activities | | | | | | |
| The context of the conversation can be unclear | | | | | | |
| The information is volatile (Dutch: vluchtig) (e.g. when you are busy you can miss important conversations) | | | | | | |
| A lack of control for the people whose conversations are overheard (e.g. people can unintentionally spread sensitive information) | | | | | | |

**A4 Comments**

## Conversations

A conversation can be overheard, listened to and participated in. In these three levels of involvement different information types are important. Here we ask you to rate the importance of a number of information items for each of these levels.

**B1 What information about a conversation is important when:**

**a) you overhear a conversation:**

| Information | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| Participants | | | | | | |
| Viewers | | | | | | |
| Location | | | | | | |
| The complete factual content | | | | | | |
| Commitment of a participant (i.e. how much someone is paying attention) | | | | | | |
| Contribution of a participant | | | | | | |
| Tone (e.g. angry, jovial, sarcastic) | | | | | | |
| Type (i.e. work related/non-work related, | | | | | | |

**Communico Survey**

| | | | | | | |
|---|---|---|---|---|---|---|
| company related/team related) | | | | | | |
| Subject | | | | | | |
| Phase (e.g. initiating, wrapping up) | | | | | | |
| Accessibility (i.e. a private conversation) | | | | | | |

**b) you listen to a conversation:**

| **Information** | -- | - | -/+ | + | ++ | **No opinion** |
|---|---|---|---|---|---|---|
| Participants | | | | | | |
| Viewers | | | | | | |
| Location | | | | | | |
| The complete factual content | | | | | | |
| Commitment of a participant (i.e. how much someone is paying attention) | | | | | | |
| Contribution of a participant | | | | | | |
| Tone (e.g. angry, jovial, sarcastic) | | | | | | |
| Type (i.e. work related/non-work related, company related/team related) | | | | | | |
| Subject | | | | | | |
| Phase (e.g. initiating, wrapping up) | | | | | | |
| Accessibility (i.e. a private conversation) | | | | | | |

**c) you participate in a conversation:**

| **Information** | -- | - | -/+ | + | ++ | **No opinion** |
|---|---|---|---|---|---|---|
| Participants | | | | | | |
| Viewers | | | | | | |
| Location | | | | | | |
| The complete factual content | | | | | | |
| Commitment of a participant (i.e. how much someone is paying attention) | | | | | | |
| Contribution of a participant | | | | | | |
| Tone (e.g. angry, jovial, sarcastic) | | | | | | |
| Type (i.e. work related/non-work related, company related/team related) | | | | | | |
| Subject | | | | | | |

**Communico Survey**

| | | | | | | |
|---|---|---|---|---|---|---|
| Phase (e.g. initiating, wrapping up) | | | | | | |
| Accessibility (i.e. a private conversation) | | | | | | |

**B2 Comments**

________________________________________________________________________

________________________________________________________________________

**B3 Please rate how important you think the following <u>actions</u> which can be performed in relation to conversations are in collaborative work:**

| Action | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| Joining a conversation | | | | | | |
| Inviting someone to join a conversation | | | | | | |
| Listening to a conversation | | | | | | |
| Dismissing other participants | | | | | | |
| Dismissing viewers | | | | | | |
| Acquiring the attention of the participants (i.e. increasing their commitment) | | | | | | |
| Notifying others (not involved in the conversation) of the conversation | | | | | | |

**B4 Comments**

________________________________________________________________________

________________________________________________________________________

## Finished Conversations

In a traditional office setting, people only have access to finished conversation based on what they recall and only when they either participated in those conversations or listened to them.

**C1 Please rate how important you think the following <u>advantages</u> of having access to finished conversations are in collaborative work:**

| Advantages | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| Having access to knowledge you might otherwise forget | | | | | | |
| Access to the technical knowledge of colleagues | | | | | | |
| Acquiring involvement (Dutch: | | | | | | |

**Communico Survey**

| | | | | | | |
|---|---|---|---|---|---|---|
| betrokkenheid) with your colleagues | | | | | | |
| Enjoying your work (e.g. overhearing a joke) | | | | | | |
| Acquiring insight in the communication structure (e.g. If someone you are looking for is absent you can contact someone he often speaks with) | | | | | | |

**C2 Comments**

**C3 Please rate how important you think the following <u>disadvantages</u> of having access to finished conversations are:**

| Disadvantages | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| It can be distracting from the current work activities | | | | | | |
| The context of the conversation can be unclear | | | | | | |
| A lack of control for the people whose conversations are overheard (e.g. people can unintentionally spread sensitive information) | | | | | | |

**C4 Comments**

## Group distribution

In Exact the distribution of the Exact Online group is not homogenous. The majority of the people work in the Delft office while several people work in Belgium, the US and from home as well. Because of this heterogeneous distribution the knowledge about others, their activities and the state of the project is not evenly distributed as well.

**D1 How big an issue are the problems caused by this non-homogenous distribution?**

[This concludes the survey for people that have not actively used Communico]

Appendix C

# Virtual Open Conversation Spaces - Discussion Guide

*This appendix depicts the document used to structure the focus group to gather insights, ideas, viewpoints and opinions of people who frequently used Communico. This focus group is discussed in chapter 6.*

## Focus Group Discussion Guide

### Objectives and information needs
To evaluate the use of Communico within Exact in particular.

### Themes
- Insight in active conversations
- Having conversations
- Insight in finished conversations
- General

### Overhearing Conversations
Discussion about the benefits and challenges of overhearing conversations

How well do you feel Communico exploits the benefits of overhearing of conversations?

- With the active conversations list and filters
- With the configurable desktop alerts?

How well do you feel Communico alleviates the challenges of overhearing conversations?

Could you give an example when overhearing a conversation was useful?

Could you give an example when overhearing a conversation was unuseful?

### Conversations
Discussion about the specific information items of a conversatioin

How well do you feel Communico shows these information items?

What could be improved?

Discuss the actions to be carried out on a conversation

How well do you feel Communico supports  these actions?

What could be improved?

## Finished Conversations

Discussion about the benefits and challenges of having access to finished conversations

How well do you feel Communico exploits the benefits of having access to finished onversations?

How well do you feel Communico alleviates the challenges of having access to finished conversations?

Could you give an example when having access to a finished conversation was useful?

Could you give an example when having access to a finished conversation was unuseful?

## General

Do you have other things you would like to mention?

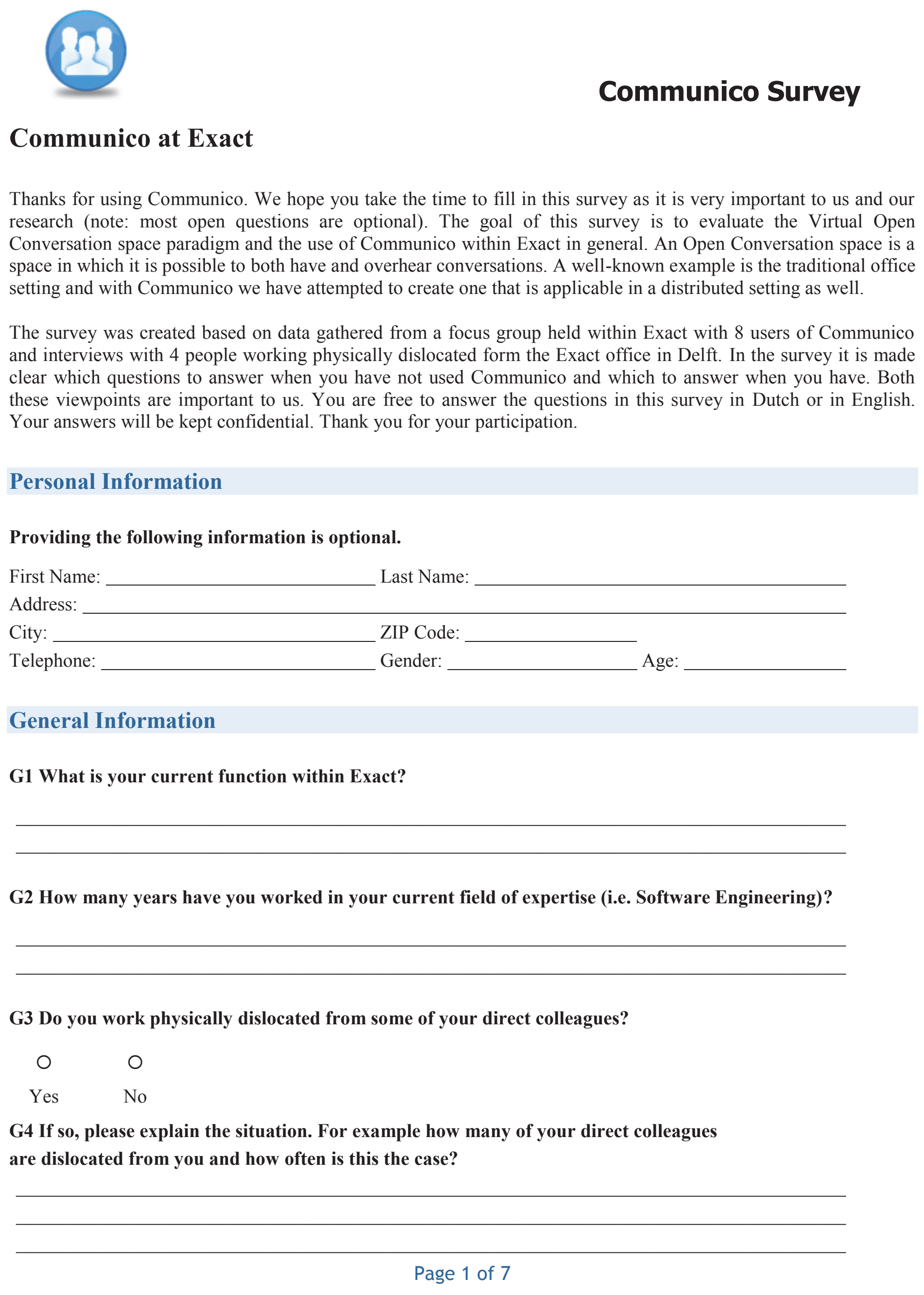
- Ideas
- Insights
- Opinions

Final Questions: Who of you would like to keep using Communico? What if all your colleagues used it as well? If not, what improvements are required to change this?

Appendix D

# Virtual Open Conversation Spaces - Survey

*This appendix depicts the document used to include the experiences of a large group of people who actively used Communico. This survey is discussed in chapter 6.*



## Communico Survey

### Communico at Exact

Thanks for using Communico. We hope you take the time to fill in this survey as it is very important to us and our research (note: most open questions are optional). The goal of this survey is to evaluate the Virtual Open Conversation space paradigm and the use of Communico within Exact in general. An Open Conversation space is a space in which it is possible to both have and overhear conversations. A well-known example is the traditional office setting and with Communico we have attempted to create one that is applicable in a distributed setting as well.

The survey was created based on data gathered from a focus group held within Exact with 8 users of Communico and interviews with 4 people working physically dislocated form the Exact office in Delft. In the survey it is made clear which questions to answer when you have not used Communico and which to answer when you have. Both these viewpoints are important to us. You are free to answer the questions in this survey in Dutch or in English. Your answers will be kept confidential. Thank you for your participation.

### Personal Information

**Providing the following information is optional.**

First Name: ______________________ Last Name: ______________________________

Address: ______________________________________________________________

City: __________________________ ZIP Code: ______________

Telephone: ______________________ Gender: _______________ Age: _____________

### General Information

**G1 What is your current function within Exact?**

_____________________________________________________________________

_____________________________________________________________________

**G2 How many years have you worked in your current field of expertise (i.e. Software Engineering)?**

_____________________________________________________________________

_____________________________________________________________________

**G3 Do you work physically dislocated from some of your direct colleagues?**

○ Yes ○ No

**G4 If so, please explain the situation. For example how many of your direct colleagues are dislocated from you and how often is this the case?**

_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________

Page 1 of 7

## Communico Survey

**G5 How often have you used Communico?**

| | | | | |
|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ |
| Never | Rarely | About one week | Several weeks | Several months |

**G6 Why did you either use or not use Communico?**

**G7 When you used Communico, how many hours a day would you generally use it?**

[This concludes the survey for people that have NOT actively used Communico]

**Communico Survey**

## Communico

We'll now ask how well you think Communico supports all the previously discussed advantages, disadvantages, information items and actions. Please note these questions are only for people that actively used Communico.

### Overhearing Conversations

**A1 Please rate how well (in your opinion) Communico enables the following advantages of overhearing conversations**

| Advantages | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| Access to the technical knowledge of colleagues | | | | | | |
| Acquiring involvement (Dutch: betrokkenheid) with your colleagues | | | | | | |
| Enjoying your work (e.g. overhearing a joke) | | | | | | |
| Acquiring insight in the communication structure (e.g. If someone you are looking for is absent you can contact someone he often speaks with) | | | | | | |
| Being able to join a conversation | | | | | | |

**A2 Please elaborate on your ratings**

______________________________

______________________________

______________________________

**A3 Please rate how well Communico copes with following disadvantages of overhearing conversations are in collaborative work:**

| Disadvantages | strongly disagree | disagree | neutral | agree | strongly agree | No opinion |
|---|---|---|---|---|---|---|
| It can be distracting from the current work activities | | | | | | |
| The context of the conversation can be unclear | | | | | | |
| The information is volatile (Dutch: vluchtig) (e.g. when you are busy you can miss important conversations) | | | | | | |
| A lack of control for the people whose conversations are overheard (e.g. people can unintentionally spread sensitive information) | | | | | | |

## Communico Survey

**A4 Please elaborate on your ratings**

____________________________________________________________
____________________________________________________________
____________________________________________________________

## Conversations

**B1 How well does Communico provide access to the following information about a conversation:**

| **Information** | -- | - | -/+ | + | ++ | **No opinion** |
|---|---|---|---|---|---|---|
| Participants | | | | | | |
| Viewers | | | | | | |
| Location | | | | | | |
| The complete factual content | | | | | | |
| Commitment of a participant (i.e. how much someone is paying attention) | | | | | | |
| Contribution of a participant | | | | | | |
| Tone (e.g. angry, jovial, sarcastic) | | | | | | |
| Type (i.e. work related/non-work related, company related/team related) | | | | | | |
| Subject | | | | | | |
| Phase (e.g. initiating, wrapping up) | | | | | | |
| Accessibility (i.e. a private conversation) | | | | | | |

**B2 Please elaborate on your ratings**

____________________________________________________________
____________________________________________________________
____________________________________________________________
____________________________________________________________
____________________________________________________________
____________________________________________________________
____________________________________________________________

## Communico Survey

**B3 Please rate how well Comuncio enables the following actions which can be performed in relation to conversations:**

| Action | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| Joining a conversation | | | | | | |
| Inviting someone to join a conversation | | | | | | |
| Listening to a conversation | | | | | | |
| Dismissing other participants | | | | | | |
| Dismissing viewers | | | | | | |
| Acquiring the attention of the participants (i.e. increasing their commitment) | | | | | | |
| Notifying others (not involved in the conversation) of the conversation | | | | | | |

**B4 Please elaborate on your ratings**

______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________

## Finished Conversations

**C1 Please rate how well Communico supports the following advantages of having access to finished conversations:**

| Advantages | -- | - | -/+ | + | ++ | No opinion |
|---|---|---|---|---|---|---|
| Having access to knowledge you might otherwise forget | | | | | | |
| Access to the technical knowledge of colleagues | | | | | | |
| Acquiring involvement (Dutch: betrokkenheid) with your colleagues | | | | | | |
| Enjoying your work (e.g. overhearing a joke) | | | | | | |
| Acquiring insight in the communication structure (e.g. If someone you are looking for is absent you can contact someone he often speaks with) | | | | | | |

## Communico Survey

**C2 Please elaborate on your ratings**

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

**C3 Please rate how well Communico enables the following <u>disadvantages</u> of having access to finished conversations:**

| **Disadvantages** | -- | - | -/+ | + | ++ | **No opinion** |
|---|---|---|---|---|---|---|
| It can be distracting from the current work activities | | | | | | |
| The context of the conversation can be unclear | | | | | | |
| A lack of control for the people whose conversations are overheard (e.g. people can unintentionally spread sensitive information) | | | | | | |

**C4 Please elaborate on your ratings**

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

## Communico Survey

### Additional feedback

To conclude this survey we'd like to ask a few general questions.

**D1 Name the 3 qualities of Communico you found most positive**

**D2 Name the 3 qualities of Communico you found most negative**

**D3 Please share any additional comments.**

Thank you for taking the time to fill out our survey. Your input is greatly appreciated.

Appendix E

# Microblogging with Mood Indicators - Interview Structure

*This appendix depicts the document used to guide the semi-structured interviews with the users of a mood based microblogging system. In these interviews we asked questions about what was unclear to us in the analysis of the microblogging usage data and we asked follow-up questions we had based on this analysis. These interviews are discussed in chapter 7.*

## Usage patterns:

### What

- What type of things do you share using WeHomer? (and what types didn't you post)
    - How did you share these things before WeHomer existed? (if so, is this better, if not, do you consider it advantageous to share these things now?) [e.g. Twitter, Skype, e-mail]
    - We noticed a relatively low amount of questions asked on WeHomer in comparison to other MicroBlogging systems. What mediums for did you use primarily for asking questions?
    - Are there other types of information you prefer to share using other communication media than WeHomer? Which ones and why?
- What type of posts do you reply to (comment)
- What type of messages on WeHomer do you find particularly helpful/insightful?
- What did you try to convey when you indicated your mood for an entry? What about a comment?
    - An example: If you comment on a sad entry, such as someone saying he is feeling ill and you comment saying get well soon, what mood did you indicate for this message?
        - Copy of the mood of whoever posted the entry
        - According to the entry (sad)
        - According to the intention of your comment (happy, cheerful)
        - According to your current actual mood, how unrelated this might be
- What did you mean to indicate with your mood when you didn't touch the slider (nothing, neutral, happy or various)
- Did you ever post something on WeHomer and regretted it later on? May I ask what? How did you handle this? Will you share this in the future using other media?

### Why

- When you check out the posts on WeHomer, what is your primary goal with this?
- When you post an entry on WeHomer, what is your primary goal with this (multiple types of entries so multiple types of answers possible)
- When you comment on an entry on WeHomer, what is your primary goal with this (multiple types of comments/entries so multiple types of answers possible)

### Who

- Are there certain people or groups of people you direct your communication to more often?
- Do you read everyone's posts equally?

### When

- Can you tell me anything about when (at what time/ at what dates/caused by what events/emotional state) you look at WeHomer?
- Same question w.r.t. placing an entry.
- When did you choose to indicate a mood and when not?

### Where

- Where do you access WeHome rmost often? For instance the mobile version at night on the couch or in public transportation

## Benefits / challenges

### Abstract

- What are in your opinion the largest advantages of using WeHomer?
- What are the largest disadvantages?

### More concrete (only ask when '*necessary*')

- What are the largest advantages of being able to see the posts/comment and associated moods of your colleagues?
- What are the largest advantages of being able to share what you are doing and your associated mood on WeHomer?
- What are the largest disadvantage of being able to see the posts/comment and associated moods of your colleagues?
- What are the largest disadvantages of being able to share what you are doing and your associated mood on WeHomer?

### So concrete it could be called leading (only ask when '*necessary*')

- Did WeHomer make you feel more connected?
- Did WeHomer make you more aware of what your colleagues were doing?
- Did you feel WeHomer caused your colleagues to be more aware of you?
- Did you worry about privacy at all?
  - Did you ever feel watched or spied on?
  - Did you feel monitored?
- Do you feel WeHomer sometimes distracted you?
- Do you feel certain people were ostracized? (buitensluiten) [For example people that participated less than others on WeHomer]

## Miscellaneous

- Do you feel a certain type of team is needed for WeHomer to be used? If so what and why?
- Has the tool been used for other purposes then its '*intended purpose*'?
- Should the tool be integrated into a comprehensive collaborative environment and if so how and why?
- How could the tool be improved?

## Overall/closing questions:

- Overall do you feel there is value in the introduction of WeHomer in IHomer?

Appendix F

# Microblogging with Mood Indicators - Coding Set

*This appendix depicts the coding set we constructed by using an iterative bootstrapping process. We used this set of codings to analyze the content of over a year of usage data of a mood based microblogging system. This gave insight in the variety of topics discussed in such a system. This coding set is discussed in chapter 7.*

The used coding set is the following. Firstly we defined four major types of codes.

1. **Nature**
2. **Form**
3. **Intention**
4. **Content**

Each of these coding categories is further divided into sub-categories and actual codes. Below we will show this subdivision for each of the four major coding categories. In these trees leaf nodes depict actual codes while non-leaf nodes depict sub-categories. Below each figure we will explain the codes and give examples. Firstly for all entries and comments we coded the nature of the message (see figure F.1). This depicts whether the content of the post can be considered positive, negative or neutral. So, for example a post stating a new assignment has landed is positive while a post about a failed build or a sick family member is negative.

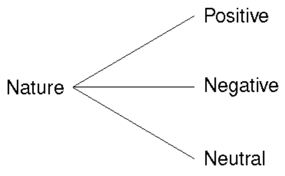


**Figure F.1:** *The nature of an entry or comment*

Secondly we also coded the form of the entries and comments (see figure F.2). With the codings we mean the following:

- **Statement**
  *An assertion*
  - **Answer**
    *Attempt to answer a question asked in a previous post*
  - **Joke**
    *Something said or done to provoke laughter or cause amusement*
  - **Compliment**
    *Expression of praise, commendation, or admiration*
  - **Best Wishes**
    *Wishing something nice to someone (such as: "good luck" or "happy birthday")*

  - **Standard**
    *A statement, but not an answer, joke, compliment or best wishes*

- **Question**
  *Attempt to illicit an answer to some question*

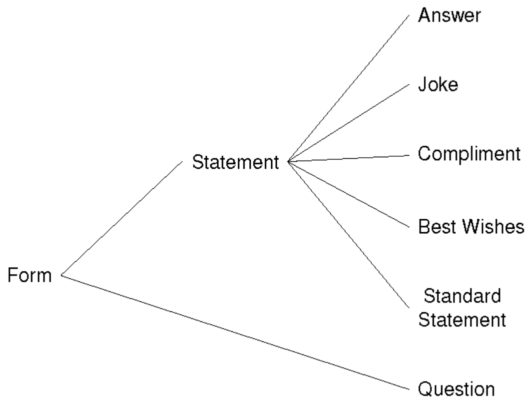


**Figure F.2:** *The form of an entry or comment*

We choose to only apply the final two coding categories (intention and content) to the entries and not the comments. We elect to do so because comments exists only in the context of the entry to which they belong. Because of this, comments are often quite brief and leave out much of this contextual information making it is infeasible for coders to consistently decide what intention and content is specifically applicable to that comment. The types of intention we distinguish are depicted in figure F.3. With these codings we mean the following:

- **Sharing personal information**
  *Intention to share information that is about the poster and his/her personal life*

- **Sharing work related information**
  *Intention to share information that is about the work of the poster*

  - **Coordination information**
    *When the intention is to coordinate with colleagues*
  - **Knowledge**
    *When the intention is to share factual knowledge*

- **Social interaction**
  *Intention to follow social protocol for making and maintaining relationships with others*

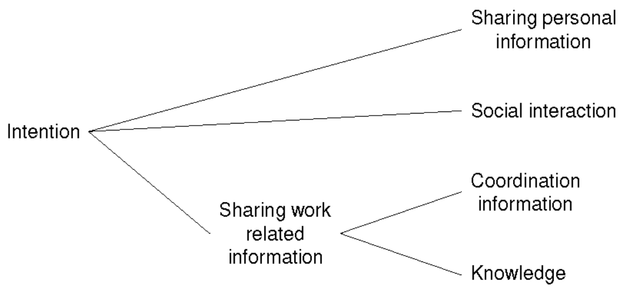

Intention
Sharing personal information
Social interaction
Sharing work related information
Coordination information
Knowledge

**Figure F.3:** *The intention of an entry*

Finally the content coding is shown in figure F.4. With these codings we mean the following:

- **Information about a person**
  - **Health**
    *The poster's health*
  - **Sentiment**
    *How the poster feels*
  - **Personal Experience**
    *Information about an experience which is not primarily work-related (Travel, Family, Scenario)*
- **Information about technology**
  - **Technical Knowledge**
    *Specialist information*
- **Information about task articulation work**
  *Information about work which is done to support the core activities of IHomer (including infrastructure and planning)*
  - **Work Planning**
    *When work will be done or is done*
  - **Work Assignment**
    *Who will do certain work (Assignment, Expertise Finding)*
  - **Supplies**
    *Information about supplies (Including for instance food)*
  - **Non-Technical Infrastructure**
    *For instance the office or office equipment*

- **Technical Infrastructure Intern**
  *For instance IHomer's timesheet application*
- **Technical Infrastructure Extern**
  *For instance DNS, Skype, IDE and phoneline*

- **Information about customer relations**
  *Information about relations with customers and the process around this*
  - **Relation**
    *Directly relating to the making and maintaining of the professional relationships with business relations*
  - **Project Commissioning**
    *About transferring finished (or partially finished work) to the customer (including things such as training)*
- **Information about entrepreneurial tasks**
  *Tasks directly related to the organization, operation and management of risk with respect to a business venture*
  - **Prospects**
    *Opportunities for new work or projects*
  - **Company Meeting**
  - **Applicants**
    *Hiring new people to work for IHomer*
  - **Invoicing**
    *About actions needed to get paid by the customers (such as sending the actual bill)*

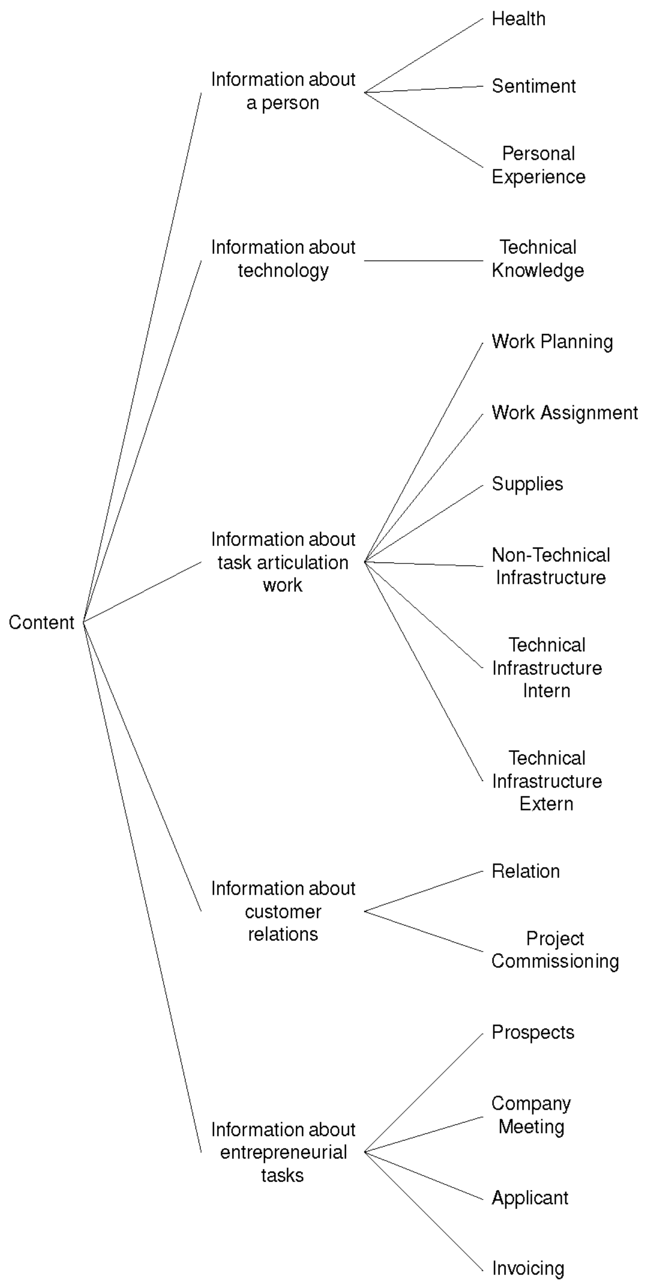

Content
Information about a person
Health
Sentiment
Personal Experience
Information about technology
Technical Knowledge
Information about task articulation work
Work Planning
Work Assignment
Supplies
Non-Technical Infrastructure
Technical Infrastructure Intern
Technical Infrastructure Extern
Information about customer relations
Relation
Project Commissioning
Information about entrepreneurial tasks
Prospects
Company Meeting
Applicant
Invoicing

**Figure F.4:** *The content of an entry*

Appendix G

# Information Needs - Questionnaire

*This appendix depicts the document used both (i) to order practices from most important to least important and distinguish between practices of which they want to be or not want to be immediately informed, and (ii) to order practices based on how disturbing an interruption would be and distinguish between practices during which they prefer to be or prefer not to be interrupted. This questionnaire is discussed in chapter 8.*

# Expert Panel

To research which information items are useful to distributed software engineers we adopt a systematic and interactive method which relies on a panel of experts. In this method, the experts first answer a questionnaire in which they indicate, (i) what kinds of conversations are interesting during software engineering activities, and (ii) when they may be interrupted by colleagues. Subsequently an interactive session is conducted in which an anonymous summary of the questionnaire is presented as an input to a discussion about unclear issues.

To structure this research we used the engineering process areas defined in the CMMI for development[1]. These engineering process areas cover the development and maintenance activities that are shared across engineering disciplines. In figure 1 the following five process areas are shown.

1. **Requirements Development (RD)**
   The purpose of Requirements Development is to elicit, analyze, and establish customer, product, and product component requirements.
2. **Technical Solution (TS)**
   The purpose of the Technical Solution process area is to select, design, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product related lifecycle processes either in isolation or in combination as appropriate.
3. **Verification (VER)**
   The purpose of Verification is to ensure that selected work products meet their specified requirements.
4. **Validation (VAL)**
   The purpose of Validation is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment.
5. **Product Integration (PI)**
   The purpose of Product Integration is to (i) assemble the product from the product components, (ii) ensure that the product, as integrated, behaves properly (e.g. possesses the required functionality and quality attributes), and (iii) deliver the product.

[1] http://www.sei.cmu.edu/reports/10tr033.pdf

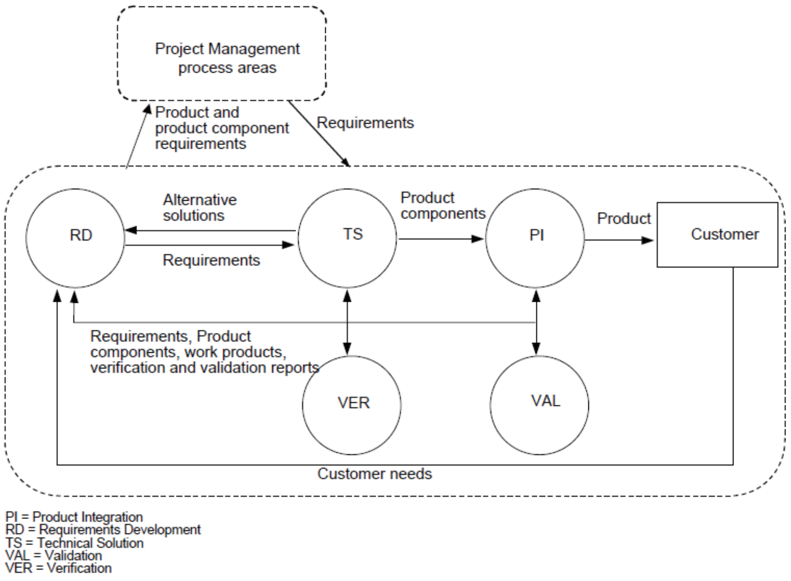

Project Management process areas
Product and product component requirements
Requirements
RD
TS
PI
Customer
Alternative solutions
Requirements
Product components
Product
Requirements, Product components, work products, verification and validation reports
VER
VAL
Customer needs
PI = Product Integration
RD = Requirements Development
TS = Technical Solution
VAL = Validation
VER = Verification

Figure 1 Engineering Process Areas

Each of these process areas has multiple sub-goals and practices and targets a specific area in the development process. We will use these process areas to structure both the questionnaire and the interactive session.

## Questionnaire

We would like you to order the practices of each of the five process areas on importance and interruptability.

Firstly, I would like you **to order the practices** (only the sub-practices [SP] not the sub-goals [SG]) **on importance regardless of your current activity**. Please start with the practice which you find most useful and conclude with the practice which you find least useful. Subsequently I would like you to split the list of practices into two parts. The first part consists of all practices of which you want to be informed regardless of your current activity while the second part consists of all practices of which you don't want to be informed of. An example of such a classification is shown on the next page in the Information Column.

Secondly, I would like you **to order the practices based on how disturbing an interruption would be while performing activities corresponding to that practice, regardless of the content of the interruption**. Please start with the practice in which an interrupt is most disturbing and conclude with the practice in which an interrupt is least disturbing. Subsequently, I would like you to split the list of practices into two parts. The first part of the list consists of all practices in which you don't want to be interrupted while the second part consists of all practices in which it is alright to be interrupted. An example of such a classification is shown in the Interruptions column in the example on the next page.

## Example

In this example we illustrate the classification of the practices of the following illustrative process area.

**SG 1. Sub-Goal 1**
- **SP 1.1. Practice 1**
- **SP 1.2. Practice 2**

**SG 2. Sub-Goal 2**
- **SP 2.1. Practice 3**
- **SP 2.2. Practice 4**

### Information

Please order the practices from most important to least important and indicate of which practices you want to be informed regardless of your current activity.

*SP Practice 3*

*SP Practice 2*

*SP Practice 4*

---

*SP Practice 1*

In this example practice 3 is considered to be the most important practice while practice 1 is considered to be the least important practice. Furthermore, it can be seen that the participant only wants to be kept up to date of conversations concerning practices 3, 2 and 4.

### Interruptions

Please order the practices based on how disturbing an interruption would be regardless of the content of the interruption and indicate whether or not you may be interrupted while performing activities corresponding to that practice.

*SP Practice 2*

---

*SP Practice 1*

*SP Practice 3*

*SP Practice 4*

In this example the participant indicated that an interruption on practice 2 is most disturbing while an interruption on practice 4 is least disturbing. Furthermore, it can be seen that the participant does not want to be disrupted while performing activities corresponding to practice 2.

## Process Area - Requirements Development

The Requirements Development process area identifies customer needs and translates these needs into product requirements. The set of product requirements is analyzed to produce a high-level conceptual solution. This set of requirements is then allocated to establish an initial set of product component requirements.

Other requirements that help define the product are derived and allocated to product components. This set of product and product component requirements clearly describes the product's performance, quality attributes, design features, verification requirements, etc., in terms the developer understands and uses.

**SG 1. Develop Customer Requirements**

De behoeften, verwachtingen, voorwaarden en interfaces van belanghebbenden worden verzameld en vertaald in klanteisen.

**SP 1.1. Elicit Needs**

Eliciteer de behoeften, verwachtingen, voorwaarden en interfaces van belanghebbenden voor alle fasen in de productlevenscyclus.

**SP 1.2. Transform Stakeholder Needs into Customer Requirements**

Transformeer behoeften, verwachtingen, voorwaarden en interfaces van belanghebbenden in geprioriteerde klanteisen.

**SG 2. Develop Product Requirements**

Klanteisen worden verfijnd en in detail uitgewerkt om de eisen voor het product en de productcomponenten te ontwikkelen.

**SP 2.1. Establish Product and Product Component Requirements**

Breng product- en productcomponenteisen tot stand die zijn gebaseerd op de klanteisen en onderhoud deze.

**SP 2.2. Allocate Product Component Requirements**

Wijs de eisen toe voor iedere productcomponent.

**SP 2.3. Identify Interface Requirements**

Identificeer de aan de interfaces gestelde eisen.

**SG 3. Analyze and Validate Requirements**

De eisen worden geanalyseerd en gevalideerd.

**SP 3.1. Establish Operational Concepts and Scenarios**

Breng operationele concepten en bijbehorende scenario's tot stand en onderhoud deze.

| | |
|---|---|
| **SP 3.2.** | **Establish a Definition of Required Functionality and Quality Attributes** |
| | Breng een definitie van vereiste functionaliteit en kwaliteitskenmerken tot stand en onderhoud deze. |
| **SP 3.3.** | **Analyze Requirements** |
| | Analyseer de eisen om ervoor te zorgen dat ze noodzakelijk en afdoende zijn. |
| **SP 3.4.** | **Analyze Requirements to Achieve Balance** |
| | Analyseer de eisen om behoeften en beperkingen van de belanghebbenden in balans te brengen. |
| **SP 3.5.** | **Validate Requirements** |
| | Valideer de eisen om te zorgen dat het resulterende product zal functioneren zoals beoogd in de eindgebruikersomgeving. |

| Information | Interruptions |
|---|---|
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |

## Process Area - Technical Solution

The Requirements Development process area supplies requirements to the Technical Solution process area, where the requirements are converted into the product architecture, product component designs, and product components (e.g. by coding, fabrication).

**SG 1. Select Product Component Solutions**

Product- of productcomponentoplossingen worden geselecteerd uit alternatieve oplossingen.

**SP 1.1. Develop Alternative Solutions and Selection Criteria**

Ontwikkel alternatieve oplossingen en selectiecriteria.

**SP 1.2. Select Product Component Solutions**

Selecteer de oplossingen voor productcomponenten op basis van selectiecriteria.

**SG 2. Develop the Design**

Er worden ontwerpen van het product of productcomponent ontwikkeld.

**SP 2.1. Design the Product or Product Component**

Ontwikkel een ontwerp voor het product of de productcomponent.

**SP 2.2. Establish a Technical Data Package**

Breng een pakket technische gegevens tot stand en onderhoud deze.

**SP 2.3. Design Interfaces Using Criteria**

Ontwerp productcomponentinterfaces met behulp van vastgestelde criteria.

**SP 2.4. Perform Make, Buy, or Reuse Analyses**

Evalueer aan de hand van vastgestelde criteria of de productcomponenten ontwikkeld, aangeschaft, of opnieuw gebruikt zouden moeten worden.

**SG 3. Implement the Product Design**

Productcomponenten en bijbehorende ondersteunende documentatie worden gerealiseerd vanuit hun ontwerpen.

**SP 3.1. Implement the Design**

Realiseer de productcomponenten vanuit de ontwerpen.

**SP 3.2. Develop Product Support Documentation**

Ontwikkel en onderhoud documentatie voor de eindgebruikers.

| Information | Interruptions |
|---|---|
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |

## Process Area - Verification

The Technical Solution process area relies on the specific practices in the Verification process area to perform design verification and peer reviews during design and prior to final build.

The Verification process area ensures that selected work products meet the specified requirements. The Verification process area selects work products and verification methods that will be used to verify work products against specified requirements. Verification is generally an incremental process, starting with product component verification and usually concluding with verification of fully assembled products.

**SG 1. Prepare for Verification**

De voorbereiding voor verificatie wordt uitgevoerd.

**SP 1.1. Select Work Products for Verification**

Selecteer de te verifiëren werkproducten en de te gebruiken verificatiemethoden.

**SP 1.2. Establish the Verification Environment**

Breng de omgeving die nodig is om verificatie te ondersteunen tot stand en onderhoud deze.

**SP 1.3. Establish Verification Procedures and Criteria**

Breng verificatieprocedures en -criteria voor de geselecteerde werkproducten tot stand en onderhoud deze.

**SG 2. Perform Peer Reviews**

Er worden peer reviews uitgevoerd op geselecteerde werkproducten.

**SP 2.1. Prepare for Peer Reviews**

Bereid voor de geselecteerde werkproducten de peer reviews voor.

**SP 2.2. Conduct Peer Reviews**

Voer peer reviews uit op geselecteerde werkproducten en identificeer probleempunten die voortkomen uit de reviews.

**SP 2.3. Analyze Peer Review Data**

Analyseer de gegevens over de voorbereiding, uitvoering en resultaten van de peer reviews.

**SG 3. Verify Selected Work Products**

Geselecteerde werkproducten worden geverifieerd tegen hun gespecificeerde eisen.

**SP 3.1. Perform Verification**

Voer verificaties uit op de geselecteerde werkproducten.

**SP 3.2. Analyze Verification Results**

Analyseer de resultaten van alle verificatie-activiteiten.

| Information | Interruptions |
|---|---|
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |

## Process Area - Validation

The Validation process area incrementally validates products against the customer's needs. Validation can be performed in the operational environment or in a simulated operational environment. Coordination with the customer on validation requirements is an important element of this process area.

The scope of the Validation process area includes validation of products, product components, selected intermediate work products, and processes. These validated elements can often require reverification and revalidation. Issues discovered during validation are usually resolved in the Requirements Development or Technical Solution process area.

**SG 1. Prepare for Validation**

De voorbereiding voor validatie wordt uitgevoerd.

**SP 1.1. Select Products for Validation**

Selecteer de te valideren producten en productcomponenten en de te gebruiken validatiemethoden.

**SP 1.2. Establish the Validation Environment**

Breng de omgeving die nodig is om validatie te ondersteunen tot stand en onderhoud deze.

**SP 1.3. Establish Validation Procedures and Criteria**

Breng procedures en criteria voor validatie tot stand en onderhoud deze.

**SG 2. Validate Product or Product Components**

De producten of de productcomponenten worden gevalideerd om hun geschiktheid te garanderen voor gebruik in hun beoogde operationele omgeving.

**SP 2.1. Perform Validation**

Voer de validatie uit op de geselecteerde producten en productcomponenten.

**SP 2.2. Analyze Validation Results**

Analyseer de resultaten van de validatieactiviteiten.

| Information | Interruptions |
|---|---|
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |
| SP | SP |

## Process Area - Product Integration

The Product Integration process area contains the specific practices associated with generating an integration strategy, integrating product components, and delivering the product to the customer.

Product Integration uses the specific practices of both Verification and Validation in implementing the product integration process. Verification practices verify the interfaces and interface requirements of product components prior to product integration. Interface verification is an essential event in the integration process. During product integration in the operational environment, the specific practices of the Validation process area are used.

**SG 1. Prepare for Product Integration**

De voorbereiding voor productintegratie wordt uitgevoerd.

**SP 1.1. Establish an Integration Strategy**

Breng een productintegratiestrategie tot stand en onderhoud deze.

**SP 1.2. Establish the Product Integration Environment**

Breng de omgeving die nodig is om de integratie van de product-componenten te ondersteunen tot stand en onderhoud deze.

**SP 1.3. Establish Product Integration Procedures and Criteria**

Breng procedures en criteria voor integratie van de productcomponenten tot stand en onderhoud deze.

**SG 2. Ensure Interface Compatibility**

Zowel de interne als externe productcomponentinterfaces zijn compatibel.

**SP 2.1. Review Interface Descriptions for Completeness**

Review interfacebeschrijvingen op dekking en volledigheid.

**SP 2.2. Manage Interfaces**

Manage interne en externe interfacedefinities, ontwerpen en wijzigingen voor producten en productcomponenten.

**SG 3. Assemble Product Components and Deliver the Product**

De geverifieerde productcomponenten worden samengevoegd en het geïntegreerde, geverifieerde en gevalideerde product wordt opgeleverd.

**SP 3.1. Confirm Readiness of Product Components for Integration**

Bevestig, voordat het product wordt samengesteld, dat elke productcomponent die nodig is voor de samenstelling van het product op de juiste wijze is geïdentificeerd, zich gedraagt volgens zijn beschrijving en dat de productcomponentinterfaces voldoen aan de interfacebeschrijvingen.

| | |
|---|---|
| **SP 3.2.** | **Assemble Product Components** |
| | Integreer de productcomponenten volgens de strategie en procedures voor productintegratie. |
| **SP 3.3.** | **Evaluate Assembled Product Components** |
| | Evalueer de geïntegreerde productcomponenten op interfacecompatibiliteit. |
| **SP 3.4.** | **Package and Deliver the Product or Product Component** |
| | Verpak het samengestelde product of de productcomponent en lever het aan de klant. |

| Information | Interruptions |
|---|---|
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |
| **SP** | **SP** |

Appendix H

# Information Needs - Post Questionnaire

*This appendix depicts the document used to ask general, process related and research related questions regarding the information needs of engineers. It is used to determine both (i) what information engineers want to know immediately, even though they are performing an activity during which they prefer not to be interrupted, and (ii) whether or not they prefer to be interrupted with information they want to know immediately, even though they are performing an activity during which they prefer not to be interrupted. This questionnaire is discussed in chapter 8.*

## General Questions

1. How many years of experience do you have, in general, with distributed software engineering?
   ______________________________________________________________________________
2. How many years of experience do you have, as a software engineer, with distributed software engineering?
   ______________________________________________________________________________
3. What is your current role (e.g. architect, tester, designer or programmer)?
   ______________________________________________________________________________
4. If you currently fulfill the role of manager, how long do you fulfill this role?
   ______________________________________________________________________________
5. Could you briefly describe the company, department and team you are currently working for (including size, location, sites etc.)?
   ______________________________________________________________________________
   ______________________________________________________________________________
   ______________________________________________________________________________
6. What kind of software development process do you use (e.g. agile, waterfall etc.)?
   ______________________________________________________________________________
7. Are you currently developing a project or a product?
   ______________________________________________________________________________

## Process Related Questions

1. How was your understanding of the practices of the five process areas after the expert panel?

| Very Good | Good | Barely Acceptable | Poor | Very Poor |
|---|---|---|---|---|
| | | | | |

2. Do the practices of the five process areas cover the complete development process? If not, please explain which aspects you are missing.
   ______________________________________________________________________________
   ______________________________________________________________________________
3. Have you reconsidered your opinion because of the arguments of other participants of the expert panel?

| Never | Rarely | Sometimes | Very Often | Always |
|---|---|---|---|---|
| | | | | |

4. Have you changed your opinion because of the peer pressure of the group of experts?

| Never | Rarely | Sometimes | Very Often | Always |
|---|---|---|---|---|
| | | | | |

# Research Related Questions

During the questionnaire and the expert panel we discussed all practices of each of the five engineering process areas. For each of these practices we discussed both, (i) whether or not you want to be directly kept up to date of information regarding that practice, and (ii) whether or not you may be interrupted while performing activities corresponding to that practice. This discussion resulted in 9 practices of which we want to be kept up to date directly and 3 practices in which we may not be interrupted (see section practices for an overview).

These outcomes introduce a contradiction since for some practices we indicated that we may not be interrupted while performing activities related to that practice. However, we also indicated that we want to be kept up to date directly of several other practices. Therefore we would like you to complete the following 12 statements in order to draw more detailed conclusions.

## Information Updates

For the following 9 practices we indicated that we want to be kept up to date directly **regardless of our current activity**. However, we also indicated that in some situations we may not be interrupted. We would like you to indicate whether or not you want to be kept up to date directly, while you are performing an **activity corresponding to a practice in which you do not want to be interrupted**.

### Example A colleague is leaving his house to go to work

*Een collega rijdt weg bij zijn huis om naar zijn werk te gaan.*

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted (e.g. taking a shower or having breakfast), and now there is new information available concerning your colleague leaving his house to go to work of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of this information?

| Yes | No |
|---|---|
| X | |

## RD SP 2.1 Establish Product and Product Component Requirements

*Breng product- en productcomponenteisen tot stand die zijn gebaseerd op de klanteisen en onderhoud deze.*

De klanteisen met betrekking tot functionaliteit en kwaliteitskenmerken kunnen uitgedrukt worden in de termen van de klant en kunnen niet-technische beschrijvingen zijn. De producteisen zijn de uitdrukking van deze eisen in technische termen die kunnen worden gebruikt voor ontwerpbeslissingen.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 2.1 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## TS SP 1.2 Select Product Component Solutions

*Selecteer de oplossingen voor productcomponenten op basis van selectiecriteria*

De toewijzingen van eisen aan productcomponenten wordt tot stand gebracht door productcomponenten te selecteren die het beste aan de criteria voldoen.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 1.2 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## TS SP 2.1 Design the Product or Product Component

*Ontwikkel een ontwerp voor het product of een productcomponent*

Het productontwerp bestaat uit twee omvangrijke fasen die elkaar in de uitvoering kunnen overlappen: globaal en gedetailleerd ontwerp. Het globale ontwerp brengt de productmogelijkheden tot stand en de productarchitectuur. Het detailontwerp definieert de structuur en mogelijkheden van de productcomponenten volledig.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 2.1 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## TS SP 2.3 Design Interfaces Using Criteria

*Ontwerp productcomponentinterfaces met behulp van vastgestelde criteria.*

De criteria voor interfaces geven vaak cruciale parameters weer die gedefinieerd of ten minste onderzocht moeten worden, om zeker te zijn van hun toepasbaarheid. Deze parameters zijn vaak specifiek voor een gegeven producttype en zijn meestal verbonden met veiligheid, beveiliging, en duurzaamheid.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 2.3 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## TS SP 3.1 Implement the Design

*Realiseer de productcomponenten vanuit de ontwerpen*

Als het ontwerp eenmaal is voltooid, dan wordt het gerealiseerd als een productcomponent. De kenmerken van die realisatie hangen af van het type productcomponent.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 3.1 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## VER SP 3.2 Analyze Verification Results

*Analyseer de resultaten van alle verificatie-activiteiten*

Werkelijke resultaten moeten worden vergeleken met de vastgestelde verificatiecriteria om te bepalen of de resultaten acceptabel zijn.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 3.2 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## VAL SP 2.2 Analyze Validation Results

*Analyseer de resultaten van de validatieactiviteiten*

De gegevens die voortvloeien uit validatietests, inspecties, demonstraties, of evaluaties worden geanalyseerd ten opzichte van de gedefinieerde validatiecriteria. In het geval van tekortkomingen, documenteren deze rapportages de mate van succes of falen en categoriseren zij de vermoedelijke oorzaak van het falen.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 2.2 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## PI SP 2.2 Manage Interfaces

*Manage interne en externe interfacedefinities, ontwerpen en wijzigingen voor producten en productcomponenten*

Interface-eisen sturen de ontwikkeling van de interfaces, die nodig zijn om productcomponenten te integreren, aan. Het managen van product- en productcomponentinterfaces start vroeg in de ontwikkeling van het product.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 2.2 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## PI SP 3.3 Evaluate Assembled Product Components

*Evalueer de geïntegreerde productcomponenten op interfacecompatibiliteit*

Deze evaluatie omvat het onderzoeken en testen van geïntegreerde productcomponenten op prestaties, geschiktheid of gereedheid, gebruikmakend van de procedures, criteria en omgeving voor productintegratie.

Currently you are performing an activity corresponding to a practice for which you indicated that you may not be interrupted, and now there is new information available concerning practice 3.3 of which we want to be kept up to date of directly. Do you, in this scenario, still want to be kept directly up to date of information regarding this practice?

| Yes | No |
|---|---|
| | |

## Interruptions

For the following 3 practices we indicated that we may not be interrupted while performing activities corresponding to that practice. However, we also indicated that we want to be kept directly up to date of information about some practices. Now we would like you to indicate, for each of these 3 practices, whether or not you may be interrupted for information you want to be kept up to date of directly.

### Example Taking a shower

*Op dit moment ben je een douche aan het nemen*

Currently you are taking a shower for which you indicated that you may not be interrupted, and now there is new information available of which you want to be kept up to date of directly (e.g. a colleague leaving his house). May you be interrupted in this scenario?

| Yes | No |
|---|---|
|  | X |

### RD SP 1.1 Elicit Needs

*Eliciteer de behoeften, verwachtingen, voorwaarden en interfaces van belanghebbenden voor alle fasen in de productlevenscyclus*

Elicitatie gaat verder dan het verzamelen van eisen, door het proactief identificeren van aanvullende eisen die niet expliciet door klanten zijn verstrekt.

Currently you are performing an activity corresponding to practice 1.1 for which you indicated that you may not be interrupted, and now there is new information available of which you want to be kept up to date of directly. May you be interrupted in this scenario?

| Yes | No |
|---|---|
|  |  |

## RD SP 3.5 Validate Requirements

*Valideer de eisen om te zorgen dat het resulterende product zal functioneren zoals beoogd in de eindgebruikersomgeving.*

Eisenvalidatie wordt vroeg in het ontwikkeltraject uitgevoerd met eindgebruikers om het vertrouwen te verkrijgen dat de eisen leiden tot een ontwikkeling die resulteert in succesvolle eindvalidatie.

Currently you are performing an activity corresponding to practice 3.5 for which you indicated that you may not be interrupted, and now there is new information available of which you want to be kept up to date of directly. May you be interrupted in this scenario?

| Yes | No |
|---|---|
|  |  |

## VER SP 3.1 Perform Verification

*Voer verificaties uit op de geselecteerde werkproducten*

Het incrementeel verifiëren van producten en werkproducten bevordert vroegtijdige ontdekking van problemen en kan resulteren in de vroegtijdige verwijdering van fouten.

Currently you are performing an activity corresponding to practice 3.1 for which you indicated that you may not be interrupted, and now there is new information available of which you want to be kept up to date of directly. May you be interrupted in this scenario?

| Yes | No |
|---|---|
|  |  |

## Remarks

1. Do you have any remarks?

# Summary

**Auto-Erecting Virtual Office Walls**
**Constructing a Virtual Office for Global Software Engineers**
- Ben van Gameren -

Software engineering is a highly collaborative activity in which knowledge, about the context in which one is working, is essential to collaborate. In the traditional co-located office setting this information is exchanged relatively passively and unobtrusively. In such an environment team members are frequently able to both see and hear each other, therefore it is relatively easy to acquire and sustain a shared understanding.

However, global software engineering is becoming increasingly common for software engineering teams, both due to the globalization of business and the rising popularity of working from home. In such a distributed environment, team members no longer share a physical work environment and are outside of sensory range of each other. Therefore it becomes infeasible to exchange information without some kind of technological support. These technological solutions, developed by the (global) software engineering community, are in general inferior to the way contextual information is shared in the co-located setting, because in comparison it (i) takes more effort, (ii) is more obtrusive, (iii) happens less frequently, and (iv) contains less information.

The goal of this dissertation is to support global software engineers with technological support for aiding them to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities. To reach this goal we use the *'industry as laboratory'* approach in which an industrial setting is used as a test environment. During our research we have closely collaborated with both IHomer and Exact Software to identify real-life global software engineering prob-

lems, develop and implement solutions for these problems, and to evaluate these solutions with experienced people from industry. To reach this goal we study three aspects of the design, implementation and evaluation of such technological support: constructing a virtual office, communicating in a virtual office, and information needs in a virtual office.

**Constructing a Virtual Office**
First, we formulate both our vision on how to provide distributed software engineers with a sufficient level of awareness and an approach to construct such technological solutions. After comparing the co-located office setting with the distributed setting we conclude that in a co-located setting awareness is spread relatively passively and unobtrusively, while it takes more effort in a distributed setting. Therefore, we propose the use of a mechanism which has the potential to regulate the available information based on the current activity of an engineer: a *'Virtual Office Wall'*. Next to formulating our vision we also present an iterative approach to develop, implement and validate such solutions. We discuss the high desirability to collaborate with companies in which distributed collaboration is common as well as the requirements such a process should fulfill. The results of these studies provide important guidelines on how best to design and implement a virtual office.

**Communicating in a Virtual Office**
Secondly, we study the value of communication in global software engineering. We start with a theoretical motivation why the overhearing of conversations of others is valuable to a distributed software engineering team. We provide a definition for a conversation, discuss the various uses conversations have in collaborative work, and define an *'Open Conversation Space'*. Subsequently, we conduct a focus group and a questionnaire in a large international software development company to explore the importance of overhearing conversations. In the 8-person focus group we ask the participants to determine the benefits of overhearing conversations, challenges of overhearing conversations, information about a conversation, actions possible on a conversation, benefits and challenges of recorded conversations. Following this, in the questionnaire among 44 participants, we determine the relative importance of these benefits, challenges, information items and possible actions. Hereafter, we perform an empirical case study to measure the value of overhearing conversations in global software engineering from actual industrial experience. To do this we present a tool which explicitly supports overhearing conversations of others: Communico and deploy it in a large international software development company. The results of this empirical study provide indications of (i) how well the benefits and challenges of having access to active conversations are exploited and alleviated, (ii) how well conversations are represented in the open conversation space, (iii) indications of how well actions on a conversation are supported, and (iv) indications of how well the benefits and challenges of having access to recorded conversations are exploited and alleviated. Based on the findings of

these empirical studies we conclude that being able to overhear conversations of colleagues in a global software engineering team is valuable.

After measuring the value of overhearing conversations, we also measure the value of microblogging with mood-indicators in global software engineering. We collect the empirical data we need by mining over a year of usage data of such a microblogging solution. In this study we code the content of all posts and comments and perform semi-structured interviews with distinctive users of this system. Based on this data we draw conclusions about what sort of topics are discussed, the impact of the introduction of such a system on distributed software teams, and the impact of distribution of the software team and team composition on the use of such a system. This study provides empirical evidence that the introduction of a mood-based microblogging solution increases team-connectedness and eases access to information that is traditionally harder to acquire.

**Information Needs in a Virtual Office**

Thirdly, we study the information needs of global software engineers. We research what information global software engineers want to know immediately and when software engineers do not mind to be interrupted with such information. To gather the empirical data needed we conduct an Estimate-Talk-Estimate study with experienced software engineers from 9 different companies. The outcomes of this study introduce a contradiction, since participants on the one hand indicate that they like to be informed immediately of several information items. On the other hand they also indicate that they prefer not to be interrupted during some activities. Therefore, we provide a look-up table which can be used to determine whether or not software engineers want to be immediately informed of an information item.

After researching what information software engineers want to know immediately and when they do not mind to be interrupted, we study the impact of automating the process of restricting the available information to that information a software engineer needs, to carry out his current activity. Therefore, we conduct a controlled experiment to study whether there is a relation between the presence of virtual office walls and the actual and perceived speed and accuracy of the work carried out by the participants. Additionally, we measure the extent in which the participants experience the presence of virtual office walls as useful. The results of this study show that virtual office walls appear to contribute to an improved awareness of co-worker synchronicity, an easier insight in the actual workload, and a more concise overview of the work performed. In turn, these improvements mostly benefit the speed of coordination and the perception regarding overall performance.

Finally, based on these (empirical) studies, we derive a set of requirements a virtual office should fulfill. This set of requirements contributes to the main goal of this dissertation, because it provides important guidelines on how best to provide global software engineers with the information they need.

# Samenvatting

**Virtuele Kantoormuren**
**Het Construeren van een Virtueel Kantoor voor Software Engineers**
- Ben van Gameren -

Het ontwikkelen van software is een collaboratieve activiteit waarin kennis over de context waarin je werkt essentieel is om samen te werken. In de traditionele kantoor omgeving wordt deze informatie passief en onopvallend uitgewisseld. In een dergelijke omgeving zijn teamleden vaak in de gelegenheid om elkaar zowel te zien als horen, daarom is het relatief eenvoudig om een gemeenschappelijk beeld te krijgen en te behouden.

Tegenwoordig komt het echter steeds vaker voor dat software gedistribueerd ontwikkeld wordt. Dit komt zowel door de globalisering van het bedrijfsleven als door de toenemende populariteit van het thuiswerken. In een dergelijke gedistribueerde omgeving delen teamleden niet langer dezelfde fysieke werk locatie en kunnen elkaar niet langer zintuiglijk waarnemen. Daardoor wordt het onmogelijk om informatie uit te wisselen zonder gebruik te maken van technologische ondersteuning. Vergeleken met een traditionele kantooromgeving heeft deze technologische ondersteuning de volgende beperkingen: het delen van informatie kost meer moeite, het delen van informatie is meer opdringerig, het delen van informatie gebeurt minder vaak, en het bevat minder informatie.

Het doel van dit proefschrift is het technologisch ondersteunen van software ontwikkelaars die gedistribueerd samenwerken door hen te helpen met het passief en onopvallend verkrijgen van contextuele informatie. Hierbij hebben we de *'industry as laboratory'* aanpak gehanteerd, waarbij een industriële omgeving wordt gebruikt als test omgeving. Tijdens ons onderzoek hebben we nauw samengewerkt met zowel IHomer als Exact Software om relevante software engineering

problemen te identificeren, om oplossingen voor deze problemen te ontwikkelen en te implementeren, en om deze oplossingen te evalueren met ervaren mensen uit het bedrijfsleven. Om het doel te bereiken bestuderen we drie aspecten van dergelijke technologische ondersteuning: het construeren van een virtueel kantoor, het communiceren in een virtueel kantoor en de informatiebehoeften in een virtueel kantoor.

**Construeren van een Virtueel Kantoor**
Ten eerste formuleren we zowel onze visie, over hoe software ontwikkelaars te helpen met het passief en onopvallend verkrijgen van contextuele informatie, als onze aanpak om dergelijke technologische oplossingen te ontwikkelen. Na het vergelijken van de traditionele kantoor omgeving met de gedistribueerde omgeving kunnen we concluderen dat in een traditionele kantoor omgeving informatie passief en onopvallend wordt uitgewisseld terwijl dit meer moeite kost in een gedistribueerde omgeving. Daarom stellen we voor een mechanisme te gebruiken dat het potentieel heeft om de beschikbare informatie te reguleren op basis van de huidige activiteit van een engineer: een *'Virtual Office Wall'*. Naast het formuleren van onze visie presenteren we ook een iteratieve aanpak om deze oplossingen te ontwikkelen, te implementeren en te valideren. We bespreken zowel het belang om samen te werken met bedrijven waarin gedistribueerd software ontwikkelen de norm is, als de eisen waaraan deze aanpak moet voldoen. De resultaten van deze studies bevatten belangrijke richtlijnen waaraan het ontwerp en de uitvoering van een virtueel kantoor moet voldoen.

**Communiceren in een Virtueel Kantoor**
Ten tweede, bestuderen we het belang van communicatie in een gedistribueerd software ontwikkelteam. Daarbij geven we een theoretische motivatie waarom het meeluisteren met gesprekken van anderen waardevol is, definiëren we een gesprek, bespreken we de verschillende toepassingen die deze gesprekken kunnen hebben in gezamenlijk werk en definiëren een *'Open Conversation Space'*. Vervolgens stellen we een focusgroep samen met deelnemers van een grote internationale onderneming, waarin we met behulp van een questionnaire het belang van het meeluisteren met gesprekken van anderen onderzoeken. De deelnemers van de focusgroep vragen we naar de voordelen en nadelen van het meeluisteren met gesprekken van anderen, nuttige informatie over een gesprek, mogelijke acties na het meeluisteren van een gesprek en de voor en nadelen van het registreren van deze gesprekken. Vervolgens bepalen we het relatieve belang hiervan in een enquête onder 44 deelnemers. Tot slot doen we een case study om het belang van het meeluisteren met gesprekken van anderen meetbaar te maken in de praktijk. Om dit mogelijk te maken presenteren we een tool die het meeluisteren met gesprekken van anderen expliciet ondersteunt: Communico. Deze oplossing is vervolgens gebruikt in een groot internationaal software bedrijf. De resultaten van dit empirisch onderzoek geven een indicatie hoe goed: (i) de voordelen van het meeluisteren met gesprekken worden benut en de nadelen worden verminderd, (ii) gesprekken wor-

den gerepresenteerd , (iii) de mogelijke acties op een gesprek worden ondersteund en (iv) de voordelen van beëindigde en geregistreerde gesprekken worden benut en de nadelen worden verminderd. De resultaten van deze empirische studies tonen aan dat de mogelijkheid om mee te luisteren met gesprekken van collega's in een gedistribueerd ontwikkelteam waardevol is.

Naast het bestuderen van het belang van meeluisteren met gesprekken van anderen in een gedistribueerd software ontwikkelteam, bestuderen we ook het belang van microblogging met stemmingsindicatoren. In deze studie hebben we, met behulp van het toegepaste microblogging systeem, de gebruiksgegevens verzameld over een periode van meer dan een jaar. Hierbij hebben we de inhoud van alle berichten en reacties gecodeerd en interviews gehouden met verschillende gebruikers van het systeem. Op basis van deze gegevens hebben we conclusies getrokken over de onderwerpen die besproken worden, de impact van de invoering van een dergelijk systeem op gedistribueerde software ontwikkel teams en de impact van de distributie en samenstelling van het software ontwikkel team op het gebruik van een dergelijk systeem. Deze studie levert empirisch bewijs dat de invoering van een stemming-gebaseerde microblogdienst de verbondenheid binnen een team verhoogt en de toegang tot vluchtige informatie makkelijker maakt.

**Informatiebehoeften in een Virtueel Kantoor**

Ten derde onderzoeken we de informatiebehoeften van gedistribueerde software ontwikkelteams. We onderzoeken welke informatie software ontwikkelaars onmiddellijk nodig hebben en wat het beste moment is om deze informatie tot zich te nemen. Om de benodigde empirische gegevens te verkrijgen voeren we een Estimate-Talk-Estimate studie uit met ervaren software ontwikkelaars van 9 verschillende bedrijven. De uitkomsten van deze studie introduceren een contradictie, aangezien de deelnemers aan de ene kant onmiddellijk willen worden ingelicht over bepaalde informatie, aan de ander kant geven zij ook aan dat ze liever niet gestoord worden bij bepaalde activiteiten. Daarom hebben we een opzoektabel gedefinieerd die gebruikt wordt om te bepalen of software ontwikkelaars onmiddellijk willen worden ingelicht over bepaalde informatie.

Hierna bestuderen we de impact van het proces, dat de beschikbare informatie beperkt tot die informatie die een ontwikkelaar nodig heeft voor het uitvoeren van zijn huidige activiteit. Hierbij voeren we een gecontroleerd experiment uit om te onderzoeken of er verband bestaat tussen de aanwezigheid van virtual office walls en de feitelijke en gepercipieerde snelheid en nauwkeurigheid van het verrichte werk. Daarnaast hebben we ook gemeten in hoeverre de deelnemers de aanwezigheid van virtual office walls als nuttig ervaren. De resultaten van deze studie tonen aan dat virtual office walls: bijdragen aan een beter inzicht in wat je collega's op dit moment aan het doen zijn, een duidelijker inzicht geven in wat er nog te doen is, een duidelijker overzicht bieden van verrichte werkzaamheden. Deze verbeteringen komen vooral ten goede aan de snelheid van coördinatie en de perceptie ten aanzien van de algemene prestaties.

Tot slot stellen we, op basis van deze (empirische) studies, een lijst van eisen samen, waaraan een virtueel kantoor moet voldoen. Deze set van eisen draagt bij aan het doel van dit proefschrift aangezien het belangrijke richtlijnen bevat over het informeren van ontwikkelaars die gedistribueerd samenwerken.

# Curriculum Vitae

Ben van Gameren was born in Rotterdam, The Netherlands, on November 13th 1985. After he finished secondary school in 2004, he started with the Bachelor Technical Informatics at the Delft University of Technology. After successfully completing his Bachelor in 2007, he continued his academic career and started with the Master Computer Science, also at the Delft University of Technology. In this program he studied the advantages and challenges of combining the agile and distributed development approaches and how technological support is best applied to deal with these. This research resulted both in his Master thesis, and a publication in the proceedings of the International Conference on Global Software Engineering. In 2009 he received his Master of Science degree, after which he joined the Software Engineering Research Group of the faculty Electrical Engineering, Mathematics and Computer Science.

As a PhD candidate he continued the interesting and promising line of research regarding how to support global software engineers with technological support for aiding them to acquire a sufficient level of awareness. He focused on what approach should be used to research this, the value of communication in global software engineering, and the information needs of global software engineers. During this period he was both employed by the Delft University of Technology and IHomer. Due to this setting he had the opportunity to experience real life global software engineering problems, to propose and implement solutions for these problems, and to evaluate these solutions. Eventually, this research resulted in this dissertation which is based on multiple peer-reviewed publications at international workshops and conferences on global software engineering.

## Education

**PhD in Computer Science** **October 2009 - October 2013**
Delft University of Technology, Delft, The Netherlands.
Dissertation title: *"Auto-Erecting Virtual Office Walls - Constructing a Virtual Office for Global Software Engineers"*

**MSc in Computer Science** **September 2007 - July 2009**
Delft University of Technology, Delft, The Netherlands.
Thesis title: *"Technical Support for Distributed Agile Development"*

**BSc in Computer Science** **September 2004 - July 2007**
Delft University of Technology, Delft, The Netherlands.

**VWO Diploma** **September 1998 - July 2004**
O.S.G. De Ring van Putten, Spijkenisse, The Netherlands.

## Work Experience

**IHomer** **January 2014 - present**
Software Engineer

**IHomer** **October 2009 - October 2013**
Researcher

**Cope IPS** **April 2007 - December 2008**
Software Engineer

**Ben van Gameren** has pursued his PhD in Software Engineering for the last four years. In his research he studies how to support global software engineers with technological support for aiding them to relatively passively and unobtrusively acquire a sufficient level of awareness for their work activities. To reach this goal he studies three important aspects of the design, implementation and evaluation of such technological support, namely: constructing a virtual office, communicating in a virtual office, and information needs in a virtual office. The results of these empirical studies, conducted in close collaboration with industry, provide valuable insights on how best to design and implement a virtual office, empirical evidence that overhearing conversations of colleagues is valuable, empirical evidence that a mood-based microblogging solution increases team-connectedness, and empirical evidence that virtual office walls increase the speed of coordination and the perception on overall performance. Based on these studies he derives a set of requirements a virtual office should fulfill. This set of requirements provides important guidelines on how best to provide global software engineers with the information they need.

ISBN: 978-94-6186-317-1

TU Delft

iHomer
thuis in it